UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

CONTROL REMOT UNIVERSAL AMB EL TEU SMARTPHONE

Memòria i Annexos

Autor: Carles Solé Grau
Director: Samir Kanaan Izquierdo

Convocatòria: Octubre de 2017

Resum

El present projecte té com a objectiu dissenyar una primera versió d'una aplicació mòbil. El nom que s'ha elegit per aquesta és *AllInOneIrRemote* i la seva finalitat és poder simular el comportament de qualsevol comandament a distància que funcioni amb infrarojos.

L'aplicació resultant permet a l'usuari poder controlar vora un miler de dispositius que són controlats via infrarojos descarregant les especificacions del protocol de comunicació des d'una base de dades online. L'aplicació també ofereix la possibilitat de simular el comportament de varis comandaments a la vegada.

Per tal de poder emetre les trames infraroges des de l'*smartphone* es fa ús, quan és possible, de l'emissor infraroig que porta incorporat el dispositiu, per tant l'aplicació només és compatible amb aquells dispositius que disposen d'aquest.

La memòria d'aquest projecte consta d'una primera part, on es descriuen totes les fases que s'han seguit per tal de crear l'aplicació: des de l'elecció de les tecnologies emprades per al seu desenvolupament fins a un manual d'usuari explicant com fer ús de l'app.

Al suport digital es pot trobar tot el codi font del projecte Qt de l'aplicació amb tots els fitxers necessaris.

Després de treballar aproximadament 4 mesos en aquest projecte, s'ha aconseguit obtenir una primera versió operativa de l'aplicació per a Android.

Resumen

El presente proyecto tiene como objetivo diseñar una primera versión de una aplicación móvil. El nombre que se ha elegido para esta es *AllInOneIrRemote* y su finalidad es poder simular el comportamiento de cualquier mando a distancia que funcione con infrarrojos.

La aplicación resultante permite al usuario poder controlar alrededor de un millar de dispositivos que se controlen mediante infrarrojos descargando las especificaciones del protocolo de comunicación de una base de datos online. La aplicación también ofrece la posibilidad de simular el comportamiento de varios mandos a la vez.

Para poder emitir esas tramas infrarrojas desde el *smartphone* se hace uso, cuando es posible, del emisor infrarrojo que tiene incorporado el dispositivo y por eso, la aplicación sólo será compatible con aquellos dispositivos que tienen uno.

La memoria de este proyecto consta de una primera parte, donde se describen todas las fases que se han seguido para crear la aplicación, desde la elección de las tecnologías usadas para su desarrollo hasta un manual de usuario explicando como hacer uso de l'app.

Al soporte digital se puede encontrar todo el código fuente del proyecto Qt de la aplicación con todos los ficheros necesarios.

Después de trabajar aproximadamente 4 meses en este proyecto, se ha conseguido obtener una primera versión operativa de la aplicación.

Abstract

The present projects aims to design a first version of a mobile application. The name that has been chosen is *AllInOneIrRemote* and it seeks to simulate the behaviour of any infrared remote control.

This app allows the user to control about a thousand devices that are controlled by infrared, downloading the specifications of the communication protocol from an online database. The application also offers the possibility to simulate the behavior of several commands at the same time.

The infrared signals that the app has to sent will be emit by the build-in infrared emitter that some devices have, so this app is just compatible with devices that have one.

The report of this projects explains all the phases that have been followed in order to create the application: from the choice of which technologies used for its development to the creation of a users manual in which its explained how to use the app.

Also note that in digital support, you can find the entire source code of the Qt project.

Finally I just would like to underline that, after working almost 4 months in this project, an operational first version of the app for Android has been successfully obtained.

Índex

Índex d'il·lustracions

1 | Introducció

1.1. Motivació

Avui en dia, molts aparells electrònics de la llar estan dotats amb un sistema per a poder-los controlar remotament. La majoria d'aquests comandaments usen infrarojos, malgrat ser una tecnologia que té més de 30 anys.

En els últims anys han aparegut tecnologies de comunicació sense fil molt més potents i amb les quals pots transmetre moltes més dades com poden ser la tecnologia Bluetooth i la Wifi. Aquestes tecnologies, però, a part de ser molt més cares d'implementar (hardware, certificats, etc), fan que els dispositius també consumeixin molta més energia, fent que l'autonomia d'un comandament que funcioni amb Bluetooth per exemple, sigui molt més petita, i és que les piles d'un comandament a distància infraroig poden arribar a durar més d'un any.

Per això, per fer una tasca «tant senzilla» com és enviar les oirdres per controlar una TV, un DVD, un equip de música, amb la tecnologia infraroja és suficient.

Degut doncs a que la tecnologia infraroja està molt estesa i encara té anys per recórrer, als darrers anys han aparegut algunes aplicacions mòbils que permeten simular un gran nombre de comandaments a distància. Personalment sempre m'havia captivat com un sol dispositiu, simplement descarregant-se una aplicació podia comportar-se com un nombre de dispositius molt elevat.

Per aquest motiu, alhora de triar aquest projecte i el fet de saber que volia realitzar una aplicació mòbil, vaig considerar oportú intentar entendre com funcionaven aquestes aplicacions que convertien un dispositiu Android en un control remot universal, i quina millor manera d'entendre com funciona una cosa que fent-la un mateix?

1.2. Objectius

L'objectiu principal d'aquest projecte és realitzar una primera versió d'una aplicació mòbil per a la plataforma Android. Aquesta aplicació ha de ser capaç de controlar un gran nombre de dispositius electrònics simulant el comportament dels seus controls remots.

Amb més detall, amb aquesta app l'usuari ha de poder:

- Seleccionar el model de control remot que vol simular.
- Mostrar el comandament a la pantalla del dispositiu mòbil.
- Emetre les mateixes trames infraroges que el comandament desitjat.
- Guardar diferents controls remots per a fer-ne ús posteriorment.
- Tenir un sistema d'usuaris per tal de poder guardar els comandaments a la xarxa
- Etc

A més, per tal de poder arribar a un gruix molt més elevat de gent, cal que l'aplicació compleixi amb uns certs requisits:

- **Interfície d'usuari amigable:** És imprescindible que la interfície d'usuari sigui *user friendly*, és a dir, que sigui neta i intuïtiva de cara l'usuari, que tingui un disseny actual, clar i agradable, etc.
- **Facilitat d'ús:** Als usuaris no el hi agrada haver de consultar un manual d'usuari per a utilitzar una aplicació mòbil, ni haver de mirar tutorials enlloc. És per això que ha de ser una aplicació molt fàcil de fer servir.
- **Fiabilitat de funcionament:** L'aplicació resultant ha de funcionar correctament amb tots els dispositius que es puguin seleccionar des de l'app, no es pot penjar ni tenir bugs evidents de cara l'usuari.

1.3. Estudi de mercat

En aquest subapartat es pretén fer un anàlisi de les opcions que avui en dia ofereix al mercat que tinguin unes característiques similars de les que volem dotar la nostra aplicació.

Fent una cerca ràpida al Google Play, es pot veure que existeixen algunes aplicacions que ofereixen unes funcionalitats similars.

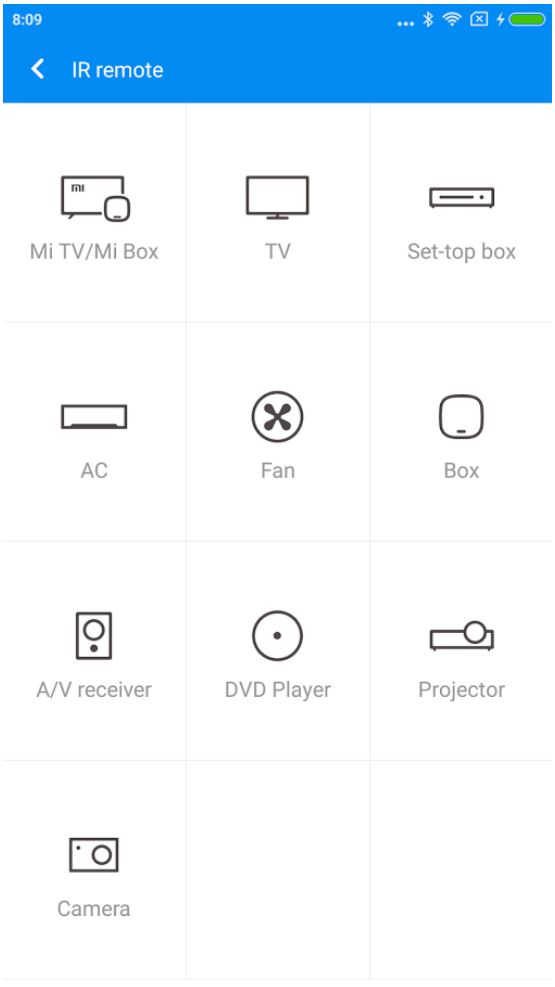
Tot seguit es fan un recull d'aquelles propostes més ben puntuades pels usuaris i que després de provar-les m'han cridat més l'atenció .

1.3.1. Mi Remote

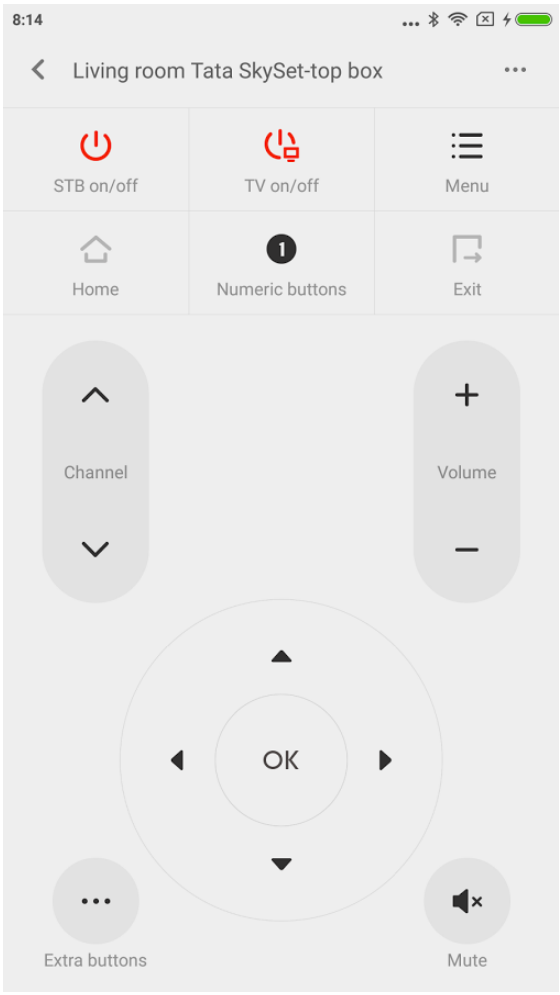
Es tracta d'una aplicació molt professional desenvolupada per l'empresa Xiaomi Inc[1], un gran fabricant xinès de dispositius mòbils. Aquest fabricant ofereix a pràcticament tots els seus nous models un transmissor infraroig i per tant els interessa que hi hagi una aplicació potent capaç de treure'n profit.

L'aplicació conta amb una gran base de dades dels protocols de comunicació i permet controlar tot tipus d'electrodomèstics que funcionin amb infrarojos. Segons la seva pàgina, alguns dels dispositius pels quals ofereixen suport són: «Aire condicionat, TV, DVD, ventilador...».

A més una de les coses que els usuaris valoren més, a part de que funcioni correctament, és el fet que no es mostren anuncis, fet que fa molt més agradable el seu ús.



Il·lustració 1: Captura de pantalla de l'aplicació Mi Remote 1/2



Il·lustració 2: Captura de pantalla de l'aplicació Mi Remote 2/2

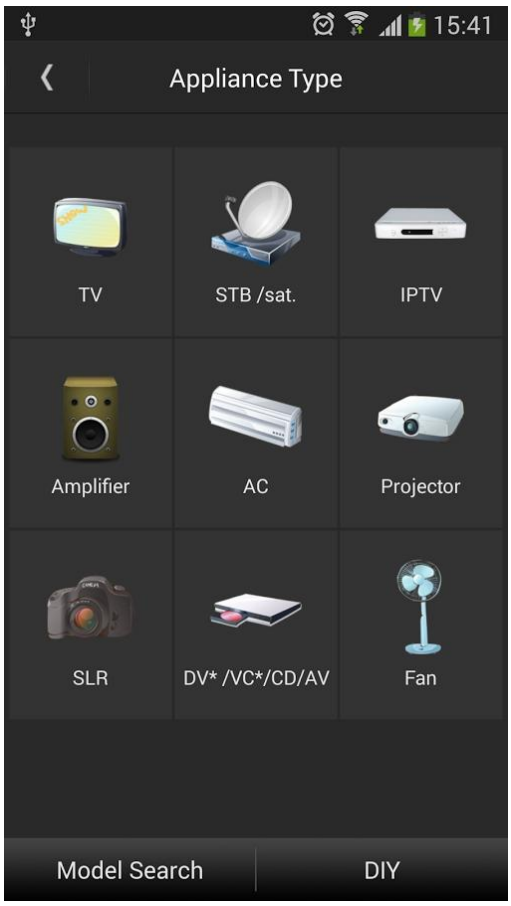
Més informació a: https://play.google.com/store/apps/details?id=com.duokan.phone.remotecontroller&hl=es

1.3.2. ZazaRemote

També és una aplicació molt completa amb una interfície molt neta i suporta un gran nombre de dispositius.. De nou, a darrere hi ha una empresa potent, no té anuncis i està molt ben valorada pels clients.

Després de fer ús d'aquesta aplicació es pot verificar que funciona correctament i, a més, en el cas que el teu dispositiu no disposi de transmissor infraroig, és compatible amb uns adaptadors infrarojos que es connecten via aux (port jack).

Aquests adaptadors es connecten pel port jack del dispositiu i el seu funcionament no és gaire òptim ja que només funcionen amb un rang de uns 10-20 cm degut a la potència que el port jack pot oferir, per tant, realment no estaríem parlant d'un control remot, ja que has d'estar pràcticament a tocar amb l'aparell que es vol controlar.



Il·lustració 3: Captura de pantalla de l'aplicació ZazaRemote 1/2

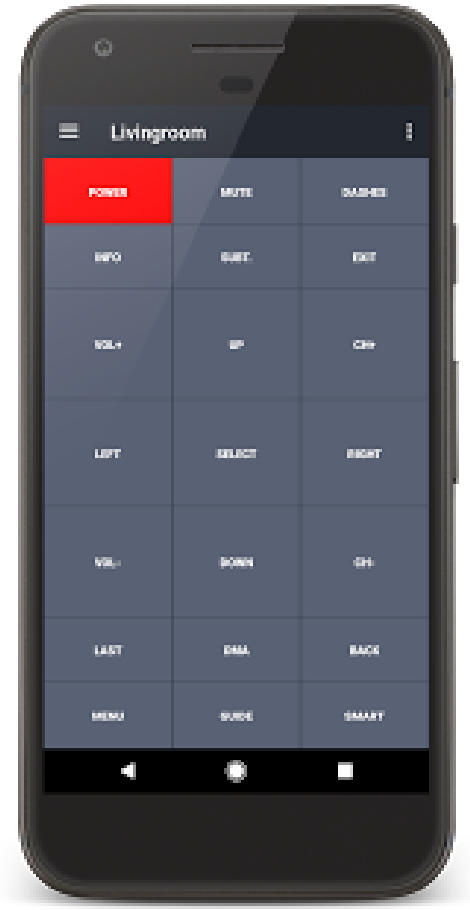
Il·lustració 4: Captura de pantalla de l'aplicació ZazaRemote 2/2

Més informació a: https://play.google.com/store/apps/details?id=com.tiqiaa.remote&hl=es_419

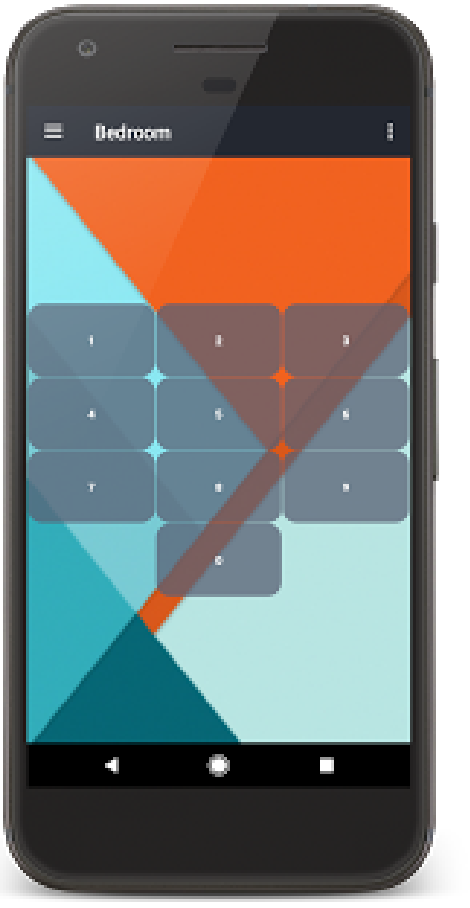
1.3.3. IR Universal TV Remote

Aplicació que només funciona amb dispositius que tenen el transmissor infraroig incorporat. L'aplicació funciona correctament amb un nombre considerable de dispositius.

Té un gran inconvenient i és que durant el seu funcionament, apareixen una gran quantitat d'anuncis molt invasius.



Il·lustració 5: Captura de pantalla de l'aplicacióIR Universal TV Remote 1/2



Il·lustració 6: Captura de pantalla de l'aplicacióIR Universal TV Remote 2/2

Més informació a: https://play.google.com/store/apps/details?id=com.freeirtv&hl=es

1.4. Nom i logotip de l'aplicació

A l'hora de definir el nom de l'aplicació , es busca que aquest sigui representatiu del què pot fer l'aplicació, per tal que quan es senti el nom, l'usuari ja pugui tenir una idea del què fa l'aplicació. Intentant seguir aquests criteris s'ha escollit el nom de *AllInOneIrRemote*.

També és important tenir un logotip que caracteritzi l'aplicació. Aquest logotip, ha de ser senzill i ha de donar a l'usuari una primera idea sobre què tracta l'aplicació. El color del logotip coincideix amb el color ambient de l'aplicació, el blau. Al logotip ha sigut dissenyat a mitjançant l'eina de programari lliure GIMP[2].

Il·lustració 7: Logotip de l'aplicació

Com es pot veure el logotip és un comandament amb el símbol infinit dins seu, fet que permet donar una idea que l'aplicació pretén ser un comandament «infinit».

2| Anàlisis

En aquest capítol es presenta la fase d'anàlisi, part inicial de qualsevol projecte de Software. Per tant es defineixen quins requisits ha de tenir l'aplicació i es dona una visió global de l'arquitectura que es vol que tingui el sistema. És molt important definir bé les bases del projecte, ja que a partir d'aquestes es construirà.

2.1. Catàleg de requisits

El catàleg de requisits és l'especificació del comportament que s'espera de l'aplicació resultant. S'ha fet una numeració dels requisits que s'han establert pel disseny i el desenvolupament de l'aplicació.

2.1.1. Requisits funcionals

Els requisits funcionals són aquells que descriuen les interaccions que els usuaris tindran amb el Software.

2.1.1.1. Gestió d'usuaris

RF1. Registre

1. L'aplicació ha d'oferir un formulari de registre per tal que l'usuari pugui introduir-hi les seves dades.
2. El sistema ha de validar les dades.
3. Ha d'aparèixer un missatge d'error en el cas que alguna de les dades del registre siguin incorrectes
4. Si les dades són correctes, el sistema guardarà les dades de l'usuari a la base de dades.

RF2. Identificació

1. Per tal de poder iniciar la sessió, cal que l'usuari s'identifiqui amb el seu nom d'usuari i la seva contrasenya.
2. El sistema permetrà o denegarà l'inici de sessió en funció de del procés d'identificació.
3. Un missatge es mostrarà si la validació de les dades no ha sigut correcte.

RF3. Tancament de sessió

1. L'aplicació ha de permetre a l'usuari tancar la sessió sempre que ho desitgi.

RF4. Eliminar Compte d'usuari

1. L'usuari ha de ser capaç d'eliminar el seu compte de la base de dades.

2.1.1.2. Gestió de comandaments

RF5. Guardar els comandaments en local

1. L'usuari ha de poder guardar els comandaments en local amb el nom que ell introdueixi.

RF6. Fer ús dels comandaments guardats en local

1. L'usuari ha de poder accedir als comandaments que ha volgut guardar sense necessitat de tornar a fer tot el procés de selecció

RF7. Eliminar comandaments de la base de dades

1. L'aplicació ha de permetre eliminar els comandaments que l'usuari ja no vol tenir emmagatzemats

RF8. Gestió al núvol dels comandaments

1. L'usuari en el cas que tingui la sessió iniciada ha de poder pujar els comandaments guardats en locals en una base de dades al núvol.

2. L'usuari en el cas que tingui la sessió iniciada ha de poder baixar-se els comandaments que té associats al seu compte al telèfon mòbil i guardar-los en local.

RF9. Mode Offline

1. En el cas que el dispositiu no tingui accés a Internet, l'usuari ha de poder executar aquells comandaments que prèviament hagi guardat en local.

2.1.1.3. Aplicació

RF10. Múltiples idiomes

1. L'aplicació ha de mostrar-se per defecte amb l'idioma que tingui el SO del *smartphone* on aquesta estigui executant-se.

2. En cas que la traducció en aquell idioma no estigui disponible, es mostraran els textos en anglès.

3. L'usuari ha de poder canviar en tot moment l'idioma de l'aplicació.

RF11. Pantalles principals

1. L'aplicació ha de constar de tres pantalles principals on l'usuari ha de poder entrar-hi en qualsevol moment durant el funcionament de l'app. Aquestes pantalles seran: gestió de comandaments, gestió d'usuaris i gestió de la configuració.

RF12. ActionBar

1. El control que permetrà moure's entre les tres pantalles principals serà una barra a l'estil «ActionBar» a la capçalera de l'aplicació.

2.1.2. Requisits no funcionals

Són requisits complementaris i/o atributs de qualitat.

RNF1. Interfície i usabilitat

1. La interfície d'usuari de l'aplicació ha de ser senzilla, atractiva i intuïtiva. El seu ús ha de ser fluid i no suposar un esforç per part de l'usuari.

RNF2. Fiabilitat de funcionament

1. L'aplicació resultant ha de funcionar correctament. Per més que una aplicació sigui fàcil de fer anar i amigable per l'usuari, si aquesta no fa la funció que s'esperava que fes, serà una aplicació que ningú voldrà fer servir.

2.2. Base de dades

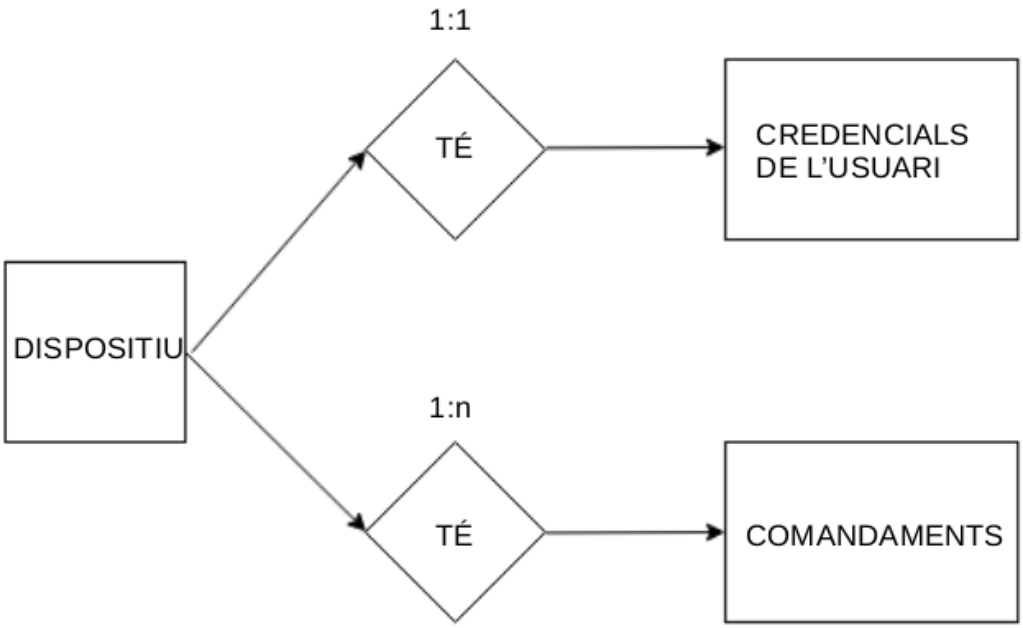
L'aplicació consta de dues bases de dades: una que s'emmagatzemarà dins el propi dispositiu i una altra d'externa.

2.2.1. Base de dades en local

Fem ús de la base de dades en local pels següents dos motius:

1. Permetre a l'aplicació mantenir la sessió iniciada amb l'últim usuari que ha entrat si aquest ho desitja: Per evitar que cada cop que entra a l'aplicació l'usuari hagi d'entrar a les seves credencials , si aquest vol que l'app arrenqui amb la sessió iniciada, a la base de dades local es guardaran les credencials de l'usuari per tal que només en arrencar l'aplicació es fagi un *log in*. Això sempre que l'usuari hagi seleccionat aquesta opció.

2. Permetre treballar mínimament sense connexió a internet: En el cas que el dispositiu mòbil no tingués accés a Internet, és interessant permetre que l'usuari pugui seguir utilitzant l'aplicació, si més no amb unes mínimes funcionalitats. En aquest cas, sense connexió, és cert que no es permetrà introduir nous comandaments, però si que es podrà treballar amb aquells que s'hagin guardat en el propi dispositiu (un nombre il·limitat de comandaments).

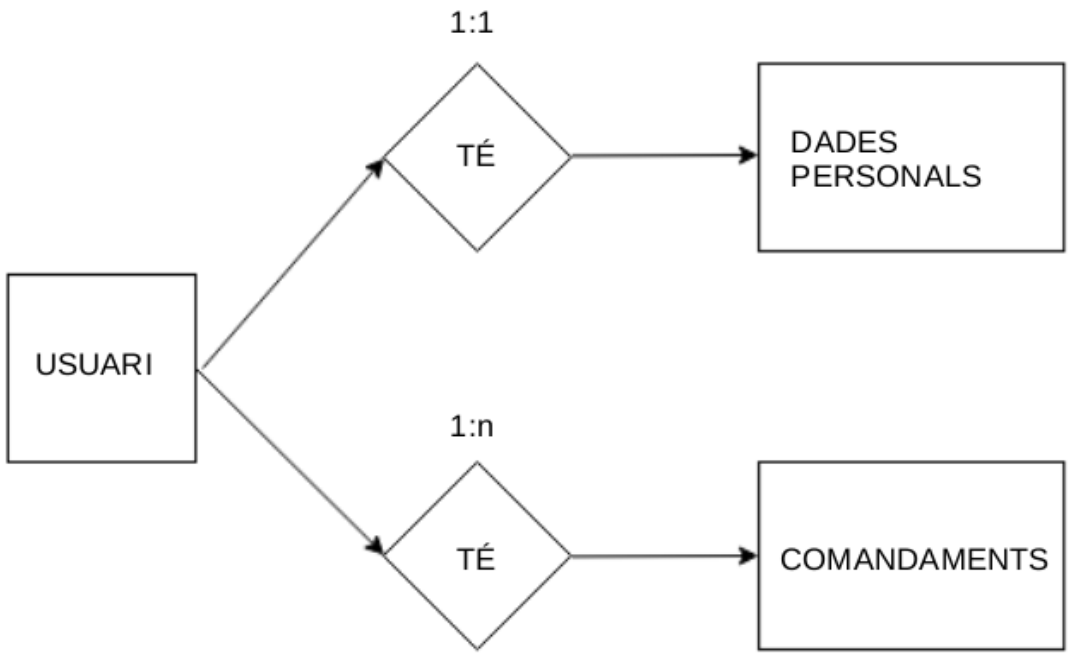


Il·lustració 8: Model entitat-relació de la base de dades en local

2.2.2. Base de dades externa

A part de la base de dades en local, l'aplicació ha d'oferir una base de dades externa, ja que es vol permetre que l'usuari tingui la possibilitat de penjar els comandaments que desitgi al seu compte per poder-los recuperar posteriorment des de qualsevol altre dispositiu. Per tant s'implementarà una base de dades externa situada a un servidor i amb la qual s'hi accedeix des de l'API del servidor.

La base de dades emmagatzemarà tota la informació dels usuaris com també els comandaments que cada un d'ells ha guardat.



Il·lustració 9: Model entitat-relació de la base de dades externa

Com es pot veure en el model entitat relació, cada usuari té unes dades personals pròpies com són el correu, la contrasenya... que podran ser consultades des de l'aplicació. A més, cada usuari podrà guardar un nombre indeterminat (n) de comandaments. Aquest model derivarà amb un disseny a la base de dades que s'explica al capítol 3.

2.3. Estudi d'API's de serveis externs

Durant el funcionament de l'app s'han d'obtenir dades de fonts externes, i aquestes s'aconsegueixen mitjançant bases de dades externes o serveis externs oferts per a terceres parts. S'ha fet ús d'una base de dades de la pàgina web http://irdb.tk/[3].

2.3.1. IRDB

Per tal d'obtenir els protocols d'infraroig per a poder controlar els diferent dispositius, cal fer ús d'una base de dades des d'on aquests es puguin consultar.

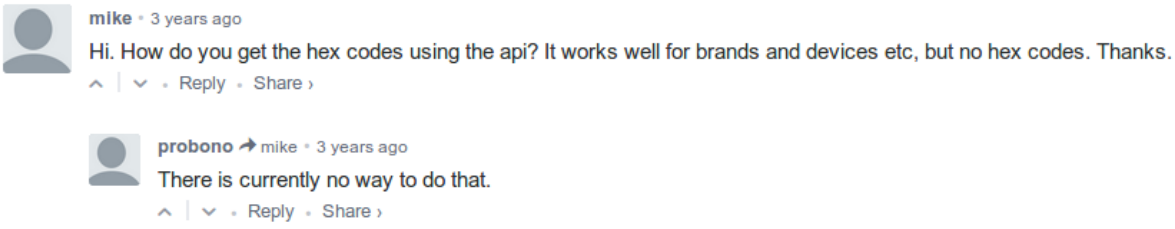
Es va plantejar la possibilitat d'afegir una d'aquestes bases de dades en local com a part de l'aplicació però es va desestimar principalment per a dos motius:

1. Augmentava significativament la mida de l'aplicació.

2. Actualitzacions molt menys constants que fent servir una base de dades online on només es consulta la informació desitjada i si aquesta s'actualitza, la informació que rebrem també ho estarà.

Després de buscar possibles solucions, es va trobar una base de dades d'una pàgina web que allotja molts protocols d'infraroig diferents. Aquesta pàgina és http://irdb.tk/[3].

Aquesta base de dades ofereix una API per tal d'interaccionar-hi[4]. Ara bé aquesta API és incomplerta ja que només et retorna les marques i models dels seus codis infrarojos que té emmagatzemats, però no ofereix la possibilitat d'obtenir aquests codis.



Il·lustració 10: Mostra del fòrum de la web irdb.tk

Com es pot veure en l'anterior figura, en un dels debats del fòrum de la web[5] la resposta de l'usuari *probono* (que es tracta de l'administrador de la web), descarta la possibilitat d'obtenir els codis mitjançant l'API.

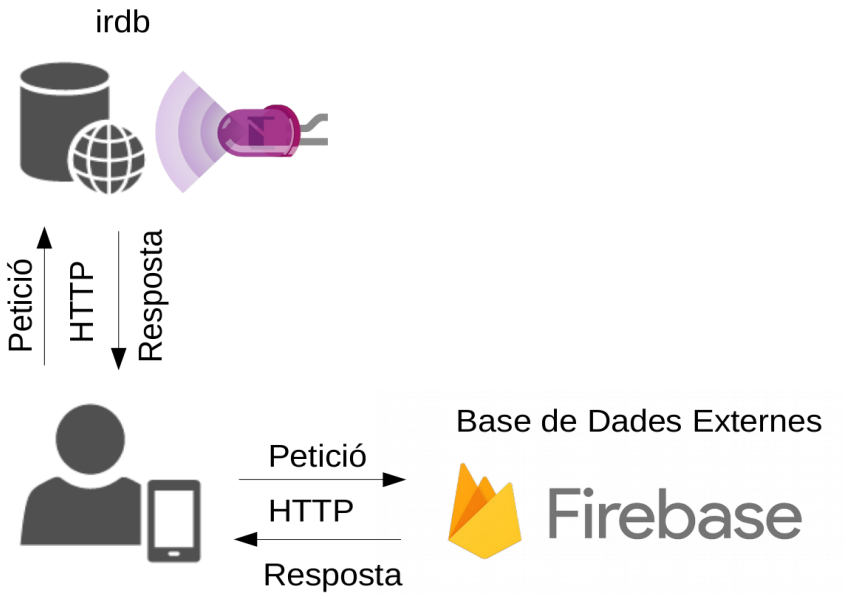
Sense els codis, doncs, aquesta base de dades era totalment inútil, per sort, després d'alguns dies d'investigació i amb l'ajuda d'un capturador d'informació de la xarxa anomenat *Wireshark*[6] es va aconseguir capturar la petició HTTP POST que fa la pàgina web per aconseguir els codis i, per tant, així s'ha pogut replicar en l'aplicació del projecte (més informació a l'apartat 4).

3 | Disseny

En aquest apartat, s'explica tot el procés de disseny de l'aplicació *AllInOneIrRremote*. El disseny que s'ha realitzat engloba tots els requisits descrits a l'apartat 2.1. El disseny proporciona una idea completa de l'aplicació desenvolupada en el projecte. A més, és el lloc on es justifiquen les decisions que s'han pres en el transcurs del desenvolupament i implementació de l'aplicació.

3.1. Arquitectura de l'aplicació

L'arquitectura de l'aplicació resultant es basa en el model client-servidor. La base de dades externa per emmagatzemar dades de l'aplicació (Firebase) i els serveis externs de terceres (base de dades amb les especificacions dels codis infrarojos: irdb) parts conformen la part del servidor, mentre que l'aplicació mòbil constitueix la part del client que accedeix al servidor mitjançant peticions HTTP.



Il·lustració 11: Arquitectura bàsica de l'aplicació

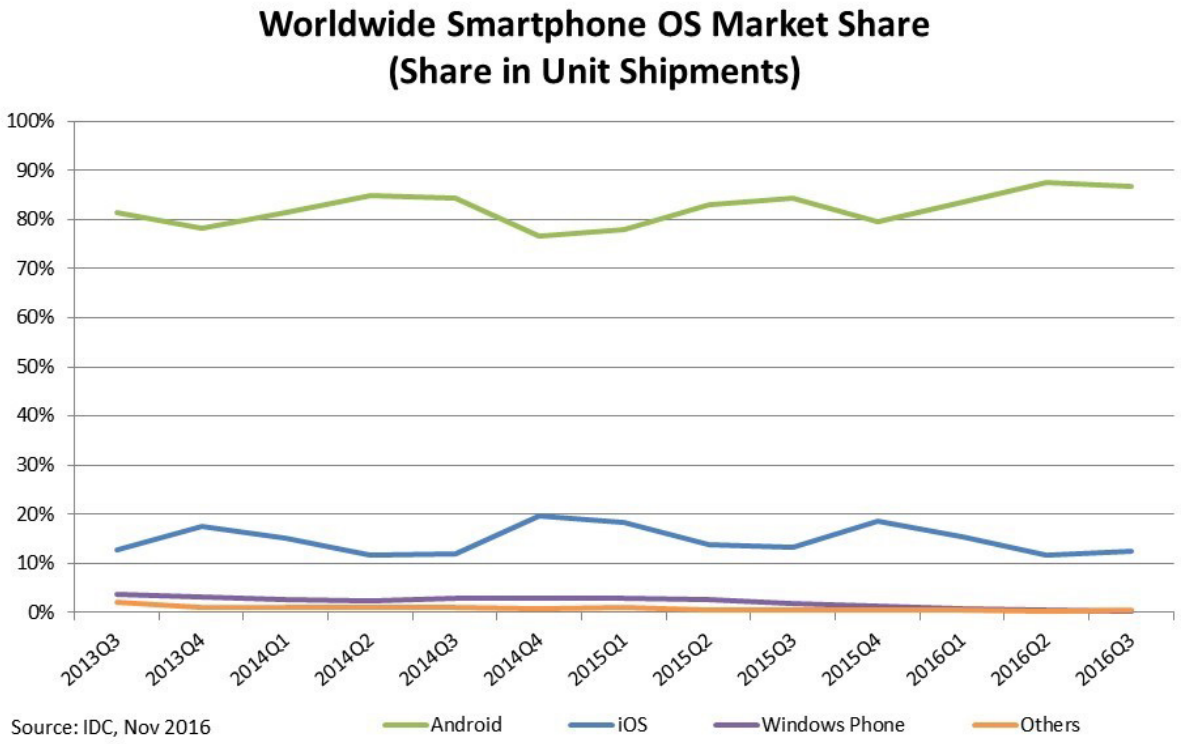
3.2. Tecnologies usades

En aquest apartat s'exposa una visió general de les tecnologies que s'han emprat pel desenvolupament de l'aplicació.

3.2.1. Plataforma de desenvolupament

A l'hora de crear una aplicació de mòbil, és important definir al principi per quin o quins sistemes operatius es busca que l'aplicació pugui funcionar.

Avui en dia, el mercat actual està dominat per les plataformes Android i iOS. Dintre d'aquestes dues i en relació al volum d'usuaris, el sistema operatiu Android està molt més estès[7], Per tant, el desenvolupament de l'aplicació es fa per a aquesta plataforma.



Il·lustració 12: Quota de mercat dels diferents sistemes operatius per a Smartphones[7]

3.2.2. Entorn de desenvolupament i llenguatge de programació

Quan es vol desenvolupar una aplicació per a Android, el llenguatge nadiu és el Java i la IDE que es proporciona des de la web d'Android i també la més utilitzada és l'anomenada *Android Studio*[8].

Existeixen, però, també moltes altres alternatives per a desenvolupar una aplicació per a Android. Per a l'aplicació d'aquest projecte s'ha optat per una d'elles: es tracta d'un conjunt d'eines que es troben recollides dintre del paraigües de l'empresa ***Qt***. La IDE que es fa servir s'anomena *QtCreator,* la qual permet crear aplicacions usant els llenguatges **QtC++** + **QML** sota la QtFramework.

Els motius principals pels quals s'ha escollit l'ús d'aquestes tecnologies són:

- Coneixement previ: Prèviament ja havia treballat amb aquesta tecnologia a l'assignatura impartida a l'EUETIB anomenada *Programació de dispositius mòbils* (PDM). A més, per un altre treball de fi de grau (d'una altra titulació) ja vaig treballar amb aquesta tecnologia i per tant, alhora de fer el desenvolupament, ja es partia d'un cert grau de coneixement.

- Eines i llenguatge multiplataforma: Qt es caracteritza per a ser un dels líders en el desenvolupament multiplataforma, tant la IDE com el llenguatge ho són. La IDE Qt Creator pot córrer sobre qualsevol sistema operatiu per a *Desktop* (Ubuntu, Windows, macOS) i, en funció del compilador que s'utilitzi, el codi programat generarà un programa que es podrà executar a qualsevol sistema operatiu (Android, iOS, Windows, Ubuntu, macOS...). Per tant, si en un futur l'aplicació es volgués portar a un altre SO, es podria reutilitzar gran part del codi, fent aquest procés molt més còmode i ràpid.

3.2.2.1. La IDE: QtCreator

Hi ha varies IDE's (Eclipse, Visual Studio...) que permeten desenvolupar programes o aplicacions amb el llenguatge QtC++ i QML afegint-els-hi plugins. Tanmateix, l'opció més recomanada és usar la pròpia IDE que ofereix Qt: **Qt Creator**[9][10]. El desenvolupament d'aquesta aplicació s'ha realitzat sobre aquesta IDE.

Qt Creator és una eina que ofereix un seguit de funcionalitats optimitzades pels llenguatges emprats:

- Editor de Text amb el suport per a C++, QML i JavaScript.

- Depurador de codi mentre s'està executant

- Gestió dels projectes i de la seva compilació

- Etc.

3.2.2.2. El Llenguatge: QtC++ i QML

QML ha sigut el llenguatge principal pel desenvolupament d'aquesta aplicació. És un llenguatge molt útil per a crear interfícies d'usuari: l'anomenat *frontend*. Malauradament, però, hi ha certes funcionalitats que no es poden dur a terme amb aquest i és per això que Qt ofereix la possibilitat d'afegir funcionalitats al programa mitjançant QtC++.

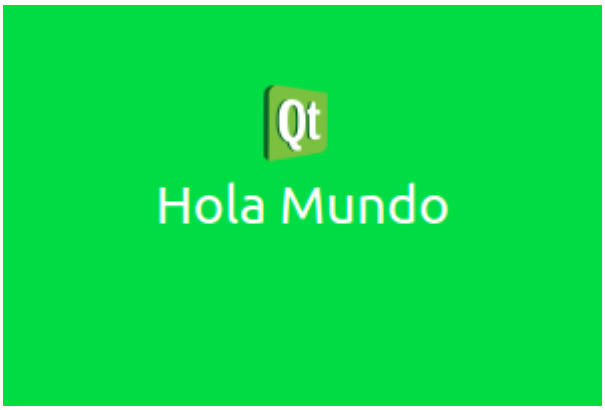
QML (Qt Meta Language) és un llenguatge de programació amb una sintaxi similar a la del JavaScript. Forma part del mòdul Qt Quick, un kit dedicat a la interfície d'usuari juntament amb el framework Qt [11][12][13].

És un llenguatge creat pensant en les aplicacions mòbils on l'entrada tàctil, les animacions fluïdes i una bona experiència amb la GUI són molt importants.

A diferencia d'altres llenguatges tradicionals com el C o el Python, QML és un llenguatge declaratiu que permet que les interfícies d'usuaris siguin descrites en termes dels seus components visuals i com ells interactuen els uns amb els altres.

Per fer-se una idea de com funciona aquest llenguatge, es mostra a continuació un petit exemple extret de la Wikipedia[14]:

```
import QtQuick 1.1
 Rectangle {
     id: canvas
     width: 300
     height: 200
     color: "#00dd44"
     Image {
         id: logo
         source: "logos/qt.png"
         x: 130
         y: 40
     }
    Text {
         id: message
         color: "white"
         text: "Hola Mundo"
         font.pointSize: 20
         font.family: "Ubuntu"
         anchors.centerIn: parent
    }
 }
```



Il·lustració 13: Exemple de interfície creada amb el llenguatge QML

Una altra avantatge que també ofereix aquest llenguatge és que el famós i estès llenguatge JavaScript està perfectament integrat:

- Es poden incorporar expressions de JavaScript com a propietats dels elements QML.
- Pots afegir funcions de JavaScript dintre el codi QML.

- Pots fer servir fitxers de JavaScript extern i importar-los al teu projecte QML.

A més, com ja s'ha comentat, quan hi ha certes funcionalitats que QML no permet fer, és possible ampliar el QML amb C++. És a dir, els elements QML poden interactuar sense cap problema amb funcions o llibreries escrites en C++. També és possible definir en C++ nous elements QML i utilitzar-los després en el codi QML.

AVANTATGES

El principal avantatge que ofereix aquesta tecnologia és el fet que es tracta d'una solució multiplataforma, és a dir, les aplicacions que nosaltres creem podran córrer sobre qualsevol superfície (Android, Linux, Windows...).

A més, és un projecte basat en el software lliure i, per tant, hi ha una gran comunitat de desenvolupadors que sovint poden resoldre els dubtes que van sorgint a mesura que es va realitzant el desenvolupament de l'aplicació.

És un llenguatge que facilita molt el procés de crear interfícies d'usuari on l'entrada tàctil, les animacions fluides i una experiència òptima de l'usuari amb l'aplicació són importants.

INCONVENIENTS

El principal inconvenient que existeix és que per desenvolupar aplicacions per a Android, aquest llenguatge no és el llenguatge nadiu i, a vegades, dur a terme certes funcionalitats especifiques per a Android és significativament més complicat. Sovint, també costa més trobar informació o exemples de codi per a funcionalitats específiques d'Android, ja que el nombre de desenvolupadors que desenvolupen aplicacions per a Android amb aquest llenguatge és molt inferior a les que ho fan amb el seu entorn i llenguatge nadiu.

3.2.3. Bases de dades

3.2.3.1. Base de dades local: *SQLite[15][16]*

La base de dades en local que s'emmagatzemarà al dispositiu per on corri l'aplicació s'implementarà en SQLite, un sistema de gestió de base de dades relacional.

El motiu pel qual s'ha optat per aquesta tecnologia és que Qt ofereix un seguit de llibreries per treballar i accedir a bases de dades escrites en aquest llenguatge des de codi escrit en QML.

3.2.3.2. Base de dades externa: *FIREBASE*[17]

Com s'ha comentat a l'apartat 2.2 fem ús d'una base de dades externa per tal de poder accedir-hi des de qualsevol dispositiu. Per tal d'implementar aquesta base de dades es fan servir els serveis de *Firebase.*

Aquesta plataforma de Google, ofereix moltes funcionalitats que ajuden als desenvolupadors a l'hora de crear la seva aplicació. De tots aquests serveis que ofereixen, pel desenvolupament de l'aplicació del projecte actual es fa servir l'autentificació d'usuaris i la base de dades en temps real.

Per a algunes plataformes, *Firebase* ofereix una SDK que permet interactuar amb els serveis d'aquesta directament des del codi amb tot un conjunt de classes i funcionalitats que la SDK importa. Malauradament per la tecnologia Qt, encara no existeix cap SDK i, per tant, es fa ús d'una REST API, la qual és accessible mitjançant peticions HTTP. [18]

Fent ús d'aquesta REST API de *Firebase*, es poden gestionar els usuaris i la base de dades amb molta més fiabilitat i robustesa que amb una base de dades creada per un desenvolupador sol, ja que a darrere d'aquests serveis hi ha una gran empresa com Google i un gran equip de desenvolupadors. En l'apartat 4 s'explicarà amb més detall com s'hi interacciona i quines peticions cal fer per a dur a terme les diferents funcionalitats.

3.2.4. Altres tecnologies usades

3.2.4.1. Sublime Text[19]

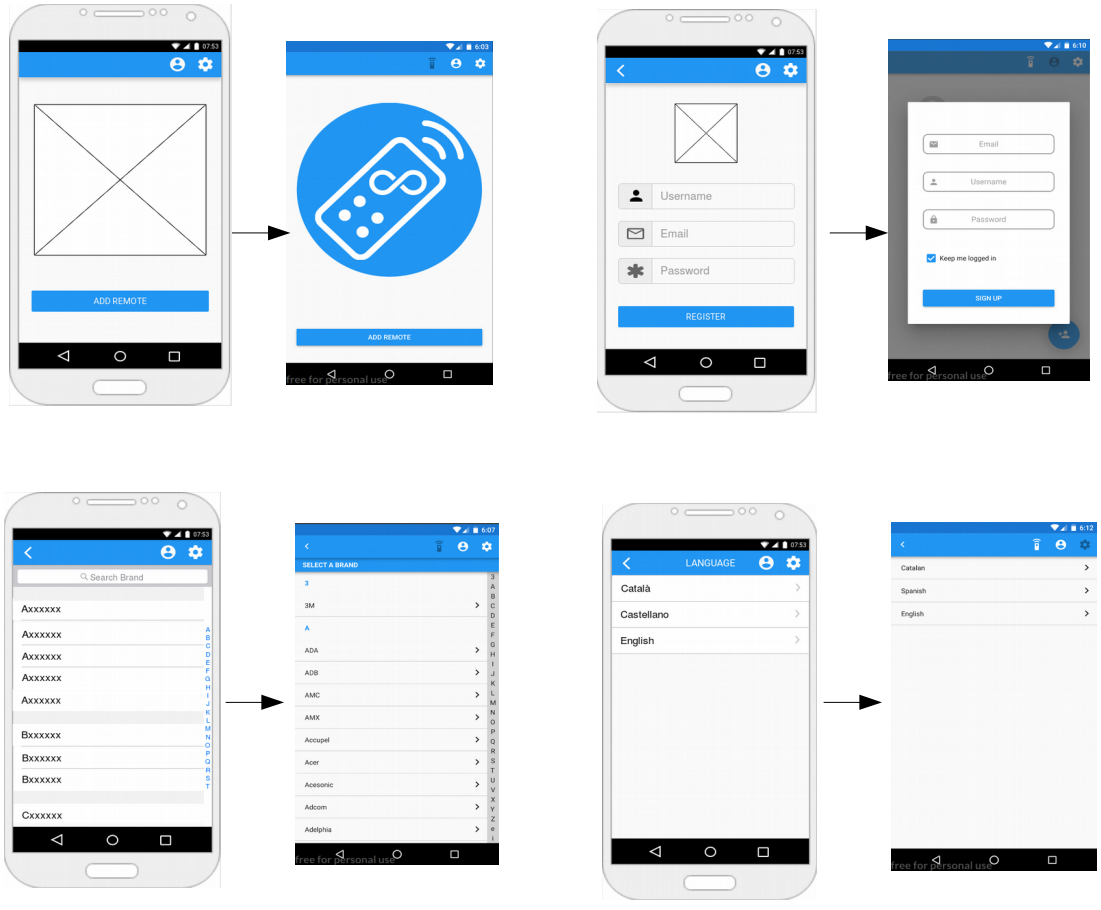
Editor de text multiplataforma. Bàsicament usat per obrir i editar els fitxers quan no es fa des de la pròpia IDE Qt Creator.

3.2.4.2. Moqups[20]

Es tracta d'una aplicació online pensada per crear esbossos de la interfície gràfica de les diferents pantalles que tindrà l'app.

Aquesta eina va ser de gran utilitat ja que permet enfocar l'esforç de decidir el disseny de les pantalles abans de programar. Així, un cop es comença a fer la programació de la part visual, només cal implementar el què prèviament s'ha decidit per a cada pantalla.

Tot seguit es poden veure algunes d'aquestes pantalles abans de programar-se (disseny Moqup) i tal i com han quedat un cop han estat programades:



Il·lustració 14: Esbossos vs Pagines reals

3.2.4.3. GenyMotion[21]

GenyMotion és un emulador d'Android bastant més ràpid que el nadiu que ofereix l'Android SDK. De gran servei a l'hora de fer córrer l'aplicació sobre l'entorn Android desitjat sense la necessitat de tenir el dispositiu físic.

3.2.4.4. GIMP[2]

Es tracta d'un programa per editar o crear imatges i/o dibuixos digitals. Està disponible sota la llicència GNU i per tant és de programari lliure. Pel present treball s'ha fet servir per a crear el logotip de l'aplicació.

3.2.4.5. Material Design [22][23]

Al llarg de tot el disseny de l'aplicació s'ha implementat el disseny *Material,* una normativa de disseny dissenyada per Google que avui en dia pràcticament totes les aplicacions Android i cada cop més webs incorporen. Al fet d'incorporar aquest disseny dóna un toc de qualitat a l'aplicació.

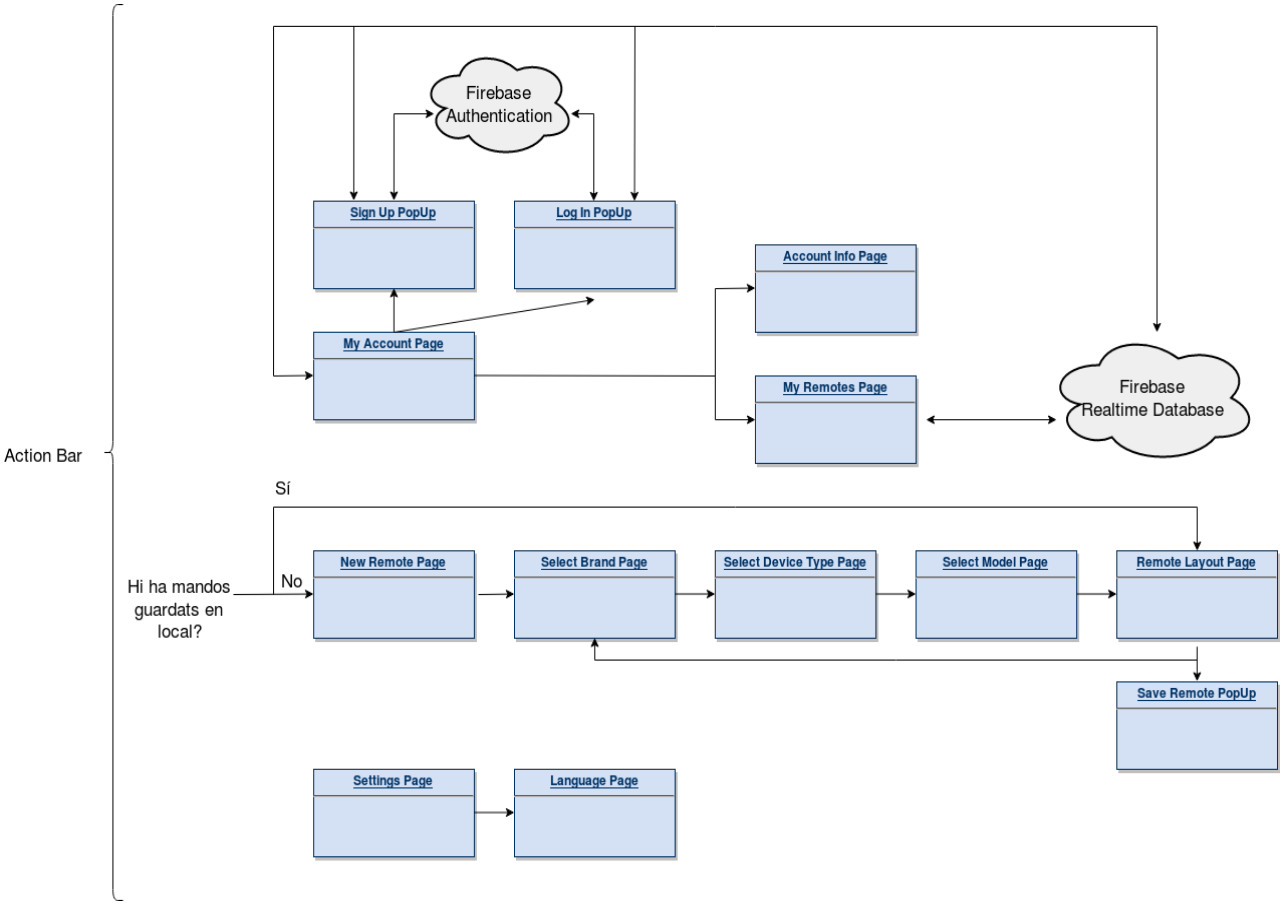
Material Design Icons[24]:

Totes les icones que es mostres a l'aplicació són extretes de la font oficial d'icones proposades per Google i segueixen el disseny proposat per a Google. En l'apartat 4 s'explica com s'han integrat aquestes icones al projecte.

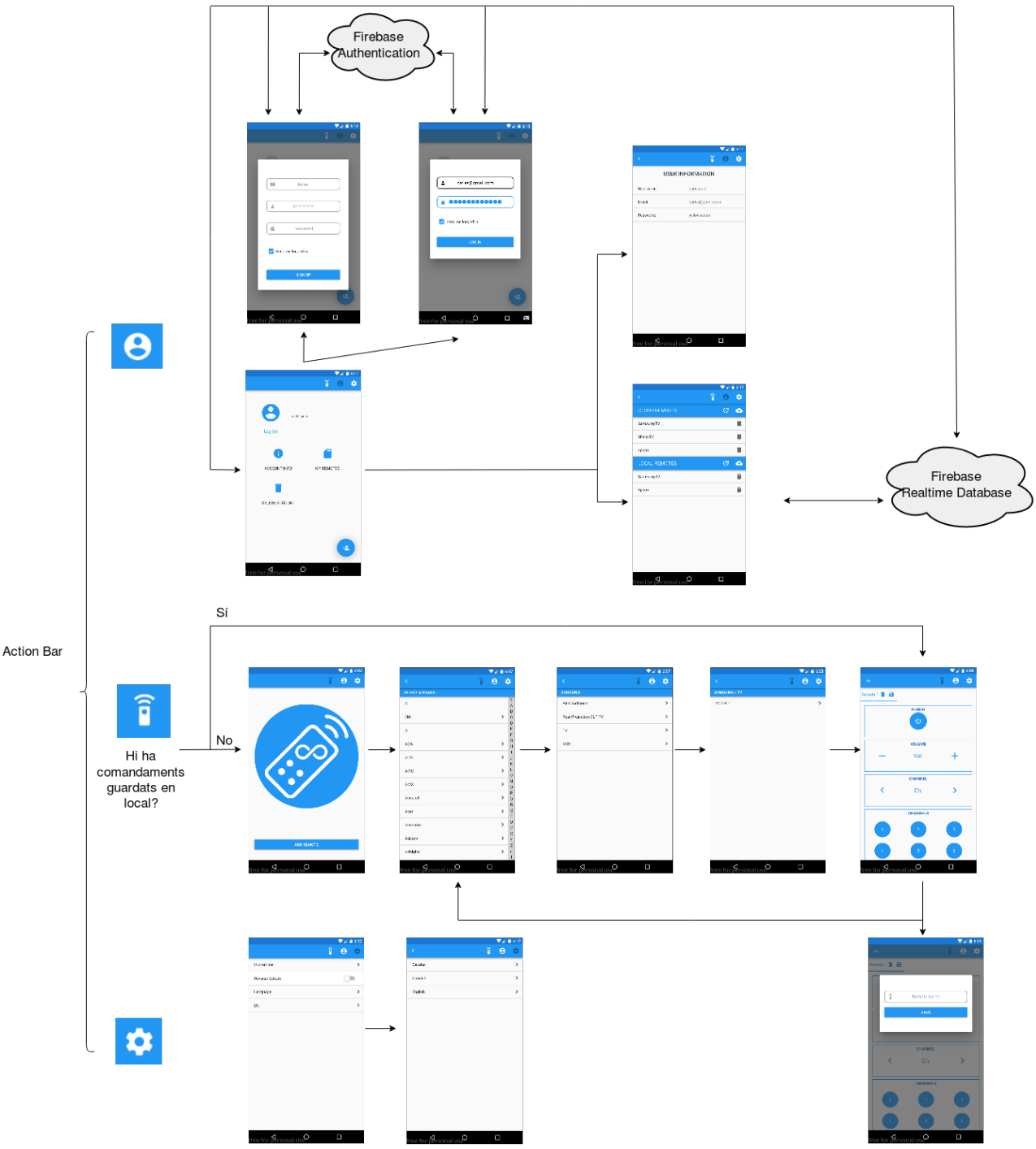
3.3. Diagrama de flux i disseny de pantalles

En aquest apartat es pretén donar una visió global del flux del funcionament de l'app on les transicions entre les principals funcions i/o pantalles quedi ben definida.

A més, en el segon gràfic, també es mostren les pantalles que es van trobant en cada moment de l'aplicació.



Il·lustració 15: Diagrama de flux de l'aplicació



Il·lustració 16: Diagrama de pantalles de l'aplicació

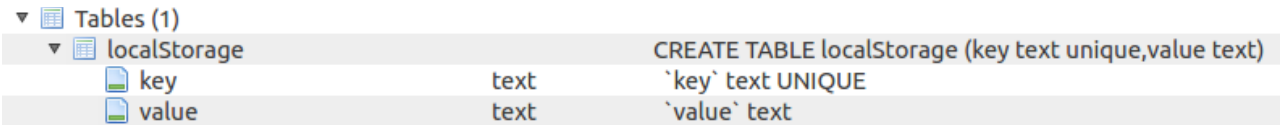
3.4. Disseny de la Base de Dades

3.4.1. Base de dades local

En aquest apartat s'explica l'estructura de la base de dades local utilitzada, que és molt senzilla ja que tal i com es mostra en el diagrama entitat-relació definit a l'apartat 2.2 només hi ha una relació de 1:n entre el dispositiu(1) i els control remots guardats (n) i una relació 1:1 entre el dispositiu(1) i les credencials de l'usuari.

El fet de treballar en SQLite suposa la necessitat de treballar en taules, ja que és la manera com es guarda la informació a la base de dades.

Ara bé, com que la base de dades a implementar era molt senzilla i a més la base de dades externa (Firebase) no treballa amb taules sinó que treballa directament amb JSON's la única taula que s'ha creat segueix més l'estructura d'un objecte JavaScript que no pas un disseny de taules relacional, ja que a la taula només hi tenim dues columnes: *key* i *value.*



Il·lustració 17: Taula emprada per a emmagatzemar la informació a la base de dades local

Així, i seguint el model entitat-relació definit a l'apartat 2.2, un cop l'aplicació estigui corrent tindrem només 3 *keys*. Aquestes seran:

- Email de l'usuari.
- Contrasenya de l'usuari
- Un string en format JSON que serà un array de tots els comandaments que l'usuari vulgui guardar en local.

3.4.2. Base de dades externa

Les funcions de gestió de l'usuari no cal que les processi la base de dades externa, ja que Firebase ja ofereix un servei que la fa.

Il·lustració 18: Mostra del taulell mostrat per a la gestió d'usuaris a Firebase

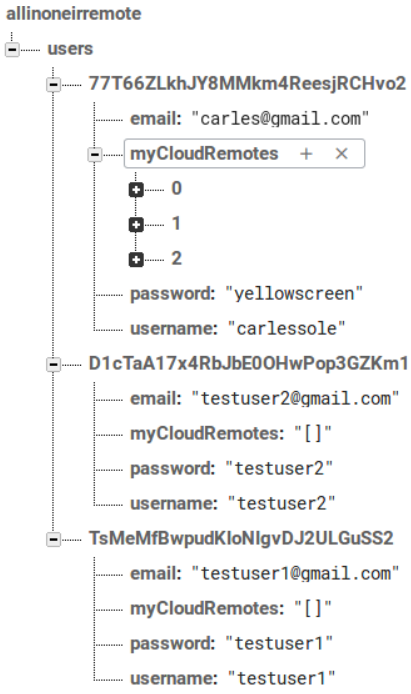
Així doncs quan l'usuari es registra, Firebase fa les gestions corresponents i omple la següent taula assignant un ID únic per a cada usuari. Posteriorment quan un usuari vol fer un inici de sessió, de nou Firebase fa les comprovacions corresponents i retorna un cert missatge a l'aplicació (èxit o fracàs del procés).

Per poder considerar Firebase com una bona opció per a fer la gestió dels usuaris, aquesta havia de poder permetre com a mínim dur a terme les següents accions:

- **Registrar Usuari.**
- **Iniciar Sessió**.
- **Eliminar usuari**.

A la documentació[25], es veu que aquestes funcionalitats estan perfectament cobertes i que a més n'ofereix moltes més com podrien ser l'autentificació de l'usuari a partir del compte de certes xarxes socials, la verificació del compte via correu, etc.

Paral·lelament a la gestió dels usuaris, tal i com s'ha comentat, es fa ús del servei de base de dades que ofereix Firebase. Quan un usuari es registra, aquesta base de dades s'omple preparant-la per tal de poder, en un futur, emmagatzemar tots els controls remots relacionats amb l'usuari. Aquesta base de dades però no treballa amb taules, sinó que treballa emmagatzemant les dades com si es tractés d'objectes de JavaScript tal i com es pot apreciar a la següent imatge:

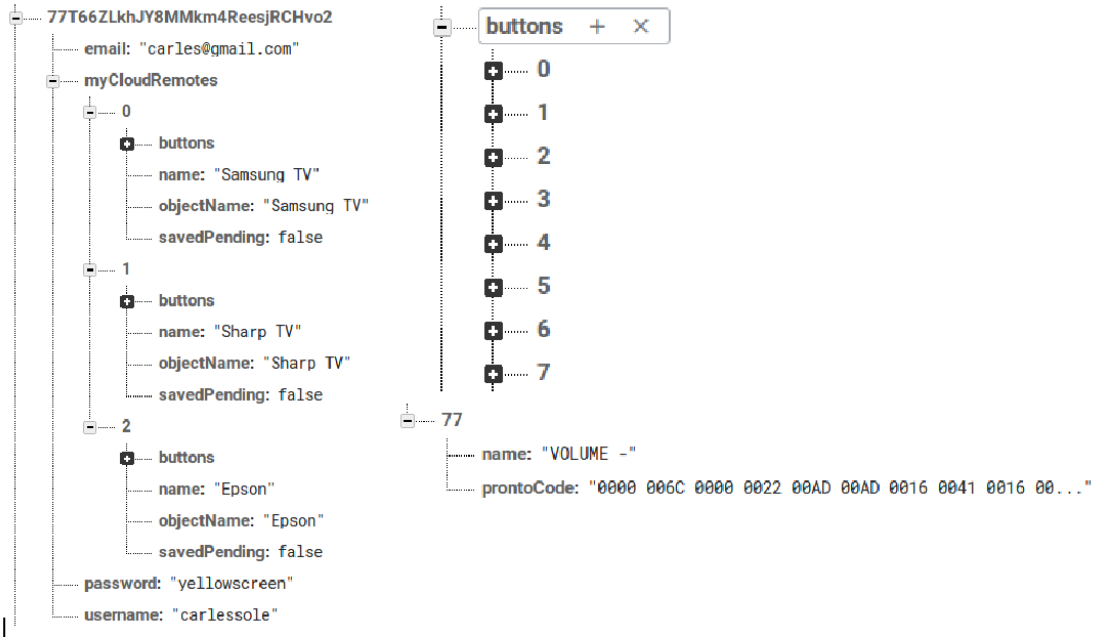


Il·lustració 19: Base de dades a Firebase 1/2

Com es veu en l'anterior imatge, de l'objecte «pare» anomenat *allinoneirremote* és pengen tots els atributs que es desitgin.

Els diferents usuaris s'identifiquen per l'ID únic que la gestió d'usuaris de Firebase els ha donat i una de les propietats de cada usuari en la base de dades externa són els comandaments: *myCloudRemotes*.

Cada comandament s'emmagatzema com un objecte que conté tota la informació que l'aplicació necessita per tal de replicar el funcionament d'aquest.



Il·lustració 20: Base de dades a Firebase 2/2

Per a validar la base de dades oferta per a Firebase aquesta havia de permetre: afegir, editar i eliminar elements. De nou, mirant la documentació de la REST API[18], es veu que totes aquestes funcionalitats estan cobertes.

4| Desenvolupament

En aquest capítol s'aborda la fase d'implementació del projecte. S'explicaran alguns detalls rellevants així com les diferents dificultats que s'han anat trobat en el procés. No s'expliquen totes les funcionalitats de l'aplicació. Si el lector té interès en mirar com està implementada alguna altra funcionalitat es pot fer mirant tot el codi font, que s'adjunta al suport digital juntament amb la memòria.

4.1. Part de l'App

En aquest subapartat s'explica el desenvolupament relacionat amb l'aplicació Android.

4.1.1. Ús d'icones Material

Les icones que van apareixent al llarg de l'aplicació s'acostumen a inserir en el codi com a imatges. Una manera molt més òptima d'entendre les icones, però, és com si aquestes es tractessin de caràcters (com podrien ser: "a","G", "?", "<"...) ja que ocupen molt menys espai i no hi ha els problemes de dimensionament associats a inserir una icona com si fos una imatge.

Per tal de poder usar les icones d'aquesta manera cal que aquestes estiguin definides dintre un fitxer .ttf, un format que s'usa per a emmagatzemar les tipografies.

Algunes de les llibreries d'icones més conegudes que existeixen actualment són *Font Awesome* [26] o les icones de la llibreria *Material* (Google)[24]. Per aquest projecte s'ha elegit fer ús de les icones Material ja que són les proposades per a Google i per tant pels dispositius Android.

El primer que cal fer és obtenir el fitxer font .ttf amb totes els icones[27]. Un cop ja el tenim cal importar-lo com a *resource* al projecte per tal de poder fer ús d'aquesta *font*. Hi ha varies maneres de fer la implementació, l'escollida s'explica a continuació.

Primer es crea un fitxer .qml anomenat **Icon.qml**:

```
Item {
    property string icon:
    property real size: 15
    property alias color: label.color
    FontLoader { id: fontMaterialIcons; source: "qrc:/Icons/MaterialIcons-Regular.ttf" }
    Label {
        id: label
        anchors.centerIn: parent
        text: icon
        font.family: fontMaterialIcons.name
        font.pointSize: size
    }
}
```

Com es pot veure en aquest tros de codi QML, per tal de dibuixar la icona, aquesta s'interpreta com si fos un text (*label.text*) i el valor que se li assigna no és més que una string (més concretament, un caràcter unicode) amb la font seleccionada amb el *FontLoader*. Aquesta string és es una propietat que se li assigna a l'invocar l'*Icon.qml*.

Per tal de dibuixar qualsevol icona a l'aplicació s'ha d'assignar aquest codi tal i com es pot veure en el següent exemple:

```
Icon{
    anchors.centerIn: parent
    icon: IconType.keyboard_arrow_left
    size: 25
}
```

A la propietat *icon*, que acaba atacant al valor del *label.text* cal assignar-li el caràcter unicode que li correspon a la icona desitjada. Es podria cada cop mirar a la documentació [24] quin codi correspon a la icona, per exemple "\ue7fe", o es pot associar el codi unicode a una variable que faciliti recordar a quin icona està associat tal i com es defineix l'objecte anomenat IconType:

```
QtObject {
    readonly property string bars: "\ue896"
    readonly property string language: "\ue894"
    readonly property string history: "\ue889"
    readonly property string settings: "\ue8b8"
    readonly property string plus: "\ue145"
    readonly property string minus: "\ue15b"
    readonly property string question: "\ue8fd"
    readonly property string check: "\ue86c"
    readonly property string times: "\ue888"
    readonly property string account_circle: "\ue853"
    readonly property string keyboard_arrow_left: "\ue314"
    readonly property string keyboard_arrow_right: "\ue315"
    readonly property string keyboard_arrow_up: "\ue316"
    readonly property string keyboard_arrow_down: "\ue313"
    readonly property string person_add: "\ue7fe"
    readonly property string person: "\ue7fd"
    readonly property string lock: "\ue897"
    readonly property string mail: "\ue158"
    readonly property string settings_remote: "\ue8c7"
    readonly property string power_settings_new: "\ue8ac"
    readonly property string info: "\ue88e"
    readonly property string delete_: "\ue872"
```

```
    readonly property string sd_card: "\ue623"
    readonly property string save: "\ue161"
    readonly property string cloud_upload: "\ue2c3"
    readonly property string cloud_download: "\ue2c0"
    readonly property string update: "\ue923"
    //....
}
```

Com es pot veure, a cada icona li correspon un d'aquests codis.

4.1.2. Suport a múltiples idiomes[28][29][30][31]

Tal i com es comenta al requisit RF10, l'aplicació ha de tenir suport per a més d'un idioma, per això cal fer ús d'un seguit d'eines que Qt ofereix, com són l'eina lupdate, lrelease i Qt Linguist.

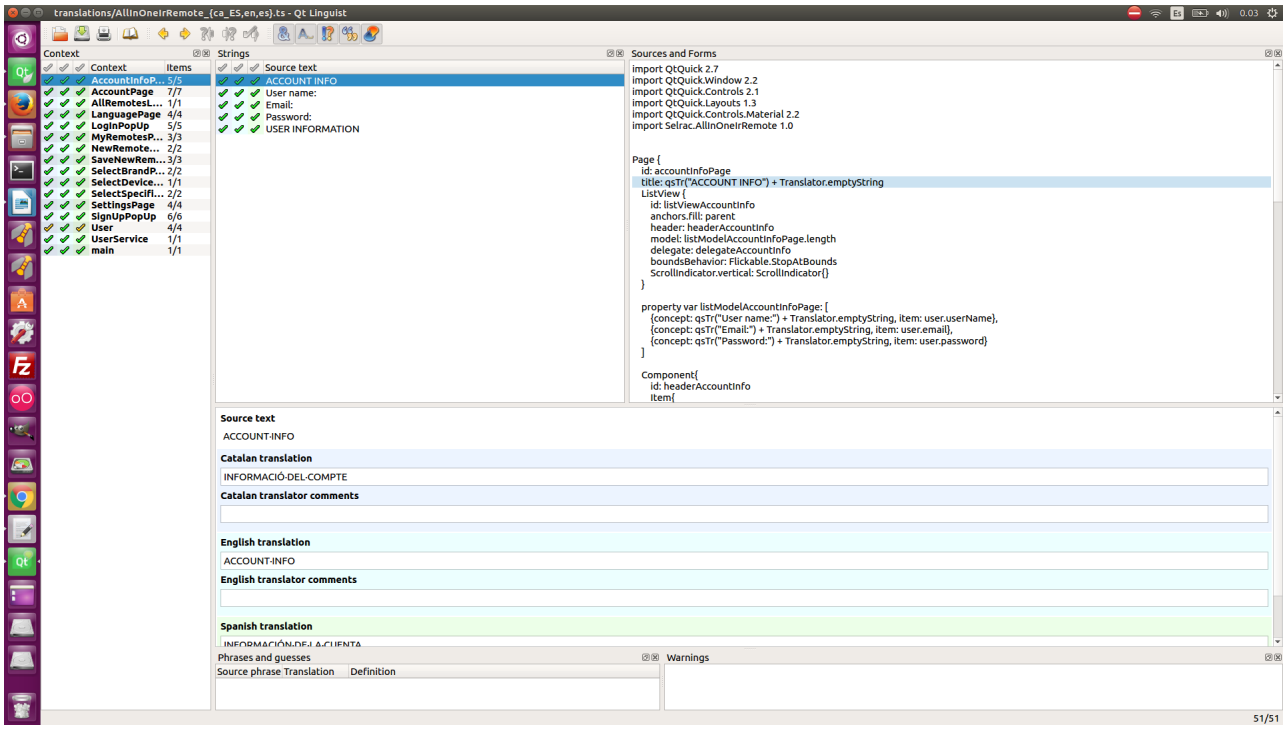
El primer que cal fer, és indicar tots aquells textos que es voldran traduir amb la funció *qsTr("text")*. Per exemple:

```
text: qsTr("DELETE ACCOUNT")
```

Amb aquesta directiva, Qt ja sap que aquest text podrà tenir múltiples traduccions i el text font és *DELETE ACCOUNT*. Un cop a tots els textos se'ls implementa la funció *qsTr()*, cal fer córrer l'eina *lupdate*, sense oblidar-se abans d'especificar al fitxer .pro quins idiomes es vol que estiguin disponibles:

```
# Supported languages
LANGUAGES = ca_ES es en
# used to create .ts files
 defineReplace(prependAll) {
     for(a,$$1):result += $$2$${a}$$3
     return($$result)
 }
# Available translations
 tsroot = $$join(TARGET,,,.ts)
 tstarget = $$join(TARGET,,,_)
 TRANSLATIONS = $$PWD/translations/$$tsroot
 TRANSLATIONS += $$prependAll(LANGUAGES, $$PWD/translations/$$tstarget, .ts)
```

Un cop executat *lupdate* es genera un fitxer amb extensió .ts per a cada idioma disponible, més un fitxer .ts amb els textos font. Aquests fitxers .ts són en realitat documents .xml que contenen tots els textos i les seves respectives traduccions. Per defecte el camp de traduccions està indefinit. És per això que cal editar cada .ts amb les traduccions de cada entrada de text. Aquesta tasca es pot fer amb qualsevol editor de textos o amb una eina que ho facilita anomenada Qt Linguist.



Il·lustració 21: Exemple de modificació dels fitxers .ts des de l'eina *QtLinguist*

Un cop ja hem definit totes les traduccions cal executar l'eina *lrelease*, la qual generarà un fitxer amb extensió .qm per a cada idioma. Aquest fitxer és el que realment carrega el Qt per a gestionar les traduccions.

Arribats aquí simplement cal indicar a l'aplicació a quin fitxer s'ha d'accedir per a cada traducció. Aquesta part no es pot fer des del llenguatge QML i per això cal fer-la mitjançant codi C++. Per això definim una classe anomenada *Translator* a la qual s'accedirà des de la banda QML.

Translator.h

```
#ifndef TRANSLATOR_H
#define TRANSLATOR_H
#include <QObject>
#include <QtQuick>
#include <QTranslator>
class Translator: public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString emptyString READ getEmptyString NOTIFY languageChanged)
public:
    Translator(QObject *parent = 0);
    Q_INVOKABLE void setLanguage(QString language);
    QString getEmptyString(void);
signals:
    void languageChanged();
private slots:
private:
    QTranslator *translator;
};
#endif // TRANSLATOR_H
```

Translator.cpp

```
#include "translator.h"
Translator::Translator(QObject *parent)    : QObject(parent)
{
    //Aquí podem inicialitzar les variables amb el seu valor per defecte
}
void Translator::setLanguage(QString language)
{
    translator = new QTranslator(this);
    if (translator->load(QLocale(language), "AllInOneIrRemote", "_",":/translations",".qm")) {
        qApp->installTranslator(translator);
    } else {
        qDebug() << "cannot load translator " << language << " check content of translations.qrc";
    }
    emit languageChanged();
}
QString Translator::getEmptyString(void)
{
    return "";
}
```

Per tal que aquesta classe sigui accessible des de la banda *QML* sense haver de fer cap instància (a la banda QML) de l'objecte *Translator* cal afegir al *Main.cpp*[32]:

```
int main(int argc, char *argv[])
{
    (...)
    Translator myTranslator;
    engine.rootContext()->setContextProperty("Translator", &myTranslator);
    (...)
}
```

Es pretén que l'app comenci a córrer amb l'idioma per defecte del SO que l'*smartphone* tingui configurat, per això cal també afegir al *Main.cpp* la següent línea:

```
int main(int argc, char *argv[])
{
    (...)
    myTranslator.setLanguage(QLocale::system().name());
    (...)
}
```

A més, l'aplicació ha de permetre a l'usuari poder escollir quin idioma vol. Com que s'ha fet la classe *Translator* accessible des de la banda QML (convertida en Objecte), aquesta tasca és molt senzilla, simplement cal fer:

```
function changeLanguage(language)
{
    Translator.setLanguage(Language);
}
```

On *Language* és una string que especifica l'idioma que es vol (ca_ES, es, en...).

Finalment i com a detall, comentar que per tal que tots els textos s'actualitzin amb el nou idioma quan es fa un canvi en *runtime*, cal fer que quelcom en la definició d'aquests canviï. És per això que la classe *Translator* conté una variable anomenada *emptyString* que s'actualitza quan es canvia el llenguatge. Aquests variable *emptyString*, tal i com anomena el seu nom, és un text buit que permet actualitzar els textos de l'aplicació:

```
text: qsTr("DELETE ACCOUNT") + Translator.emptyString
```

Cal afegir aquesta propietat a tots els textos que es volen actualitzar al canviar l'idioma de l'aplicació.

4.1.3. Crear nous objectes QML des de Qt C++ [33][34]

Com s'explica a l'apartat 3.2.2.2, és possible crear objectes que corrin en C++ i accedir-hi des de la banda QML. Aquest recurs és molt interessant per ampliar i complementar les funcionalitats del llenguatge QML.

Per tal de poder-ho fer, primer cal crear la classe (tal com s'explica a la documentació de Qt[34]) que defineix l'objecte QML:

StatusBar.h:

```
#ifndef STATUSBAR_H
#define STATUSBAR_H
#include <QObject>
#include <QColor>
#ifdef Q_OS_ANDROID
#include <QtAndroid>
#endif
class StatusBar : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QColor color READ color WRITE setColor)
    Q_PROPERTY(bool available READ isAvailable CONSTANT)
public:
    explicit StatusBar(QObject *parent = nullptr);
    QColor color() const;
    void setColor(const QColor &color);
    bool isAvailable() const;
private:
    QColor m_color;
};
#endif // STATUSBAR_H
```

StatusBar.cpp:

```
#include "statusbar.h"
StatusBar::StatusBar(QObject *parent) : QObject(parent)
{
}
QColor StatusBar::color() const
{
    return m_color;
}
void StatusBar::setColor(const QColor &color)
{
    Q_UNUSED(color);
    if (!isAvailable())
        return;
#ifdef Q_OS_ANDROID
    QtAndroid::runOnAndroidThread([=]() {
    QAndroidJniObject::callStaticMethod<void>("cat/selrac/StatusBar",
                                              "setColor",
                                              "(I)V",
                                              color.rgba());
     });
#endif
}
bool StatusBar::isAvailable() const
{
#ifdef Q_OS_ANDROID
    return QtAndroid::androidSdkVersion() >= 21;
#else
    return false;
#endif
}
```

Concretament aquesta classe interactua directament amb codi Java (codi natiu Android), tal i com es pot veure quan es crida *QandroidJniObject::callStaticMethod*. De moment obviarem aquesta funció i s'explicarà al següent punt.

Un cop generada la classe, cal afegir al fitxer main.cpp:

```
int main(int argc, char *argv[])
{
    (...)
    qmlRegisterType<StatusBar>("Selrac.AllInOneIrRemote", 1, 0, "StatusBar");
    (...)
}
```

Fet aquest pas, l'objecte *StatusBar* ja es pot instanciar com un objecte des de la banda QML com es faria amb un objecte estàndard de QML com `Rectangle{}` (important importar però en aquest cas `Selrac.AllInOneIrRemote`):

```
import Selrac.AllInOneIrRemote 1.0
(...)
StatusBar {
   id: androidStatusBar
   color: Material.color(primaryColor, Material.Shade700)
}
```

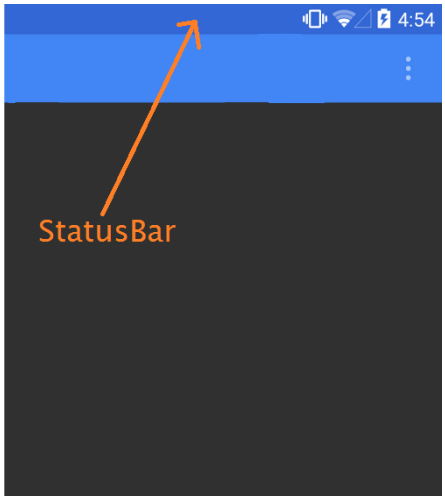
4.1.4. Cridar funcions de Java des de QtC++: Modificar StatusBar[35][36][37][38]

Quan es programa en Qt, des de la banda QtC++, és possible fer pràcticament qualsevol cosa, sobretot si es treballa sobre un entorn *Dektop*. Ara bé, en el cas que es programi per a Android, com que el llenguatge nadiu per desenvolupar per a aquesta plataforma és el Java, hi ha certes funcionalitats que encara no han estat traduïdes en llibreries per a C++.

Quan això succeïx, Qt permet cridar codi Java per executar aquestes funcionalitats. Pel present projecte, ha calgut invocar funcions de Java en dues situacions: enviar trames infraroges des de l'emissor integrat al dispositiu i canviar el color de l'*status bar* d'Android.

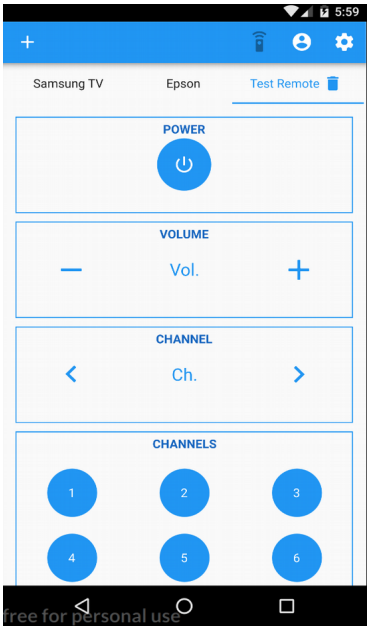
Respecte l'enviament de trames infraroges se'n farà incís més endavant. Per a explicar com invocar funcions en Java des de C++, es posarà com exemple el segon cas.

Des de QT i/o QML no hi ha cap classe (QtC++) o objecte(QML) que es refereixi a l'*status bar* d'Android.



Il·lustració 22: StatusBar al sistema Android

Per tant quan es fa córrer una aplicació programada en Qt, l'*statusbar* es mostra de color negre, ja que és el seu aspecte per defecte:



Il·lustració 23: StatusBar sense configurar, color per defecte

Com ja s'ha explicat, per modificar el color de la *statusbar* cal fer-ho des de codi Java. Concretament la classe Java utilitzada que crida el mètode és la següent:

StatusBar.java:

```
package cat.selrac;
import android.graphics.Color;
import android.view.Window;
import android.view.WindowManager;
public class StatusBar extends MyActivity
{
    public StatusBar()
    {
    }
    public static void setColor(int color)
    {
        Window window = qtActivity.getWindow();
        // clear FLAG_TRANSLUCENT_STATUS flag:
        window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);
        // add FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS flag to the window
        window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BACKGROUNDS);
        // finally change the color
        window.setStatusBarColor(color);
    }
}
```

No s'entrarà a analitzar què fa cada part d'aquest codi Java ja que ja està explicat a les referències [35][36]. El que es vol mostrar és com poder accedir a aquesta funció des de C++.

Per accedir des de C++ a codi Java cal fer servir el *Qt Android Extras*, concretament la classe *QAndroidJniObject*. Aquestes classes només estan disponibles quan es compila per a Android, per això quan es fan servir sempre aniran precedides d'una directiva del precompilador. *#ifdef Q_OS_ANDROID*. En el següent codi en C++, es veu com se'n fa ús:

StatusBar.h:

```
#ifndef STATUSBAR_H
#define STATUSBAR_H
#include <QObject>
#include <QColor>
#ifdef Q_OS_ANDROID
#include <QtAndroid>
#endif
class StatusBar : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QColor color READ color WRITE setColor)
    Q_PROPERTY(bool available READ isAvailable CONSTANT)
public:
    explicit StatusBar(QObject *parent = nullptr);
    QColor color() const;
    void setColor(const QColor &color);
    bool isAvailable() const;
private:
    QColor m_color;
};
#endif // STATUSBAR_H
```

StatusBar.cpp:

```
#include "statusbar.h"
StatusBar::StatusBar(QObject *parent) : QObject(parent)
{
}
QColor StatusBar::color() const
{
    return m_color;
}
void StatusBar::setColor(const QColor &color)
{
    Q_UNUSED(color);
    if (!isAvailable())
        return;
#ifdef Q_OS_ANDROID
    QtAndroid::runOnAndroidThread([=]() {
    QAndroidJniObject::callStaticMethod<void>("cat/selrac/StatusBar",
                                              "setColor",
                                              "(I)V",
                                              color.rgba());
     });
#endif
}
bool StatusBar::isAvailable() const
{
#ifdef Q_OS_ANDROID
    return QtAndroid::androidSdkVersion() >= 21;
#else
    return false;
#endif
}
```

De tota aquesta classe, la part de codi que més ens importa és la següent funció:

```
void StatusBar::setColor(const QColor &color)
{
    Q_UNUSED(color);
    if (!isAvailable())
        return;
#ifdef Q_OS_ANDROID
    QtAndroid::runOnAndroidThread([=]() {
    QAndroidJniObject::callStaticMethod<void>("cat/selrac/StatusBar",
                                              "setColor",
                                              "(I)V",
                                              color.rgba());
     });
#endif
}
```

Amb més detall analitzem la funció:

```
  QAndroidJniObject::callStaticMethod<void>("cat/selrac/StatusBar",
                                            "setColor",
                                            "(I)V",
                                            color.rgba());
```

El mètode *callStaticMethod* de la classe *QAndroidJniObject* permet cridar un mètode escrit en Java. La manera de crida la funció i passar els seus paràmetres és una mica complexa.

El primer paràmetre: cat/selrac/StatusBar fa referència a la classe Java que conté la funció i del seu *path* des del directori de referència *src*.

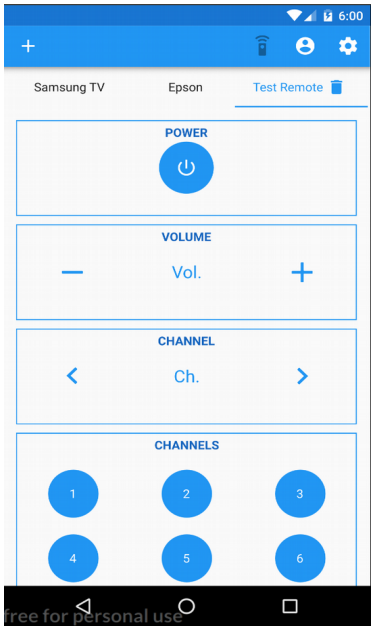
El segon paràmetre: setColor és la funció en particular que s'invoca.

El tercer: (I)V, és una anotació una mica confosa pròpia de la classe `QAndroidJniObject`. Es refereix als paràmetres i al retorn de la funció anterior: *setColor*. Els valors dintre els parèntesis és el tipus de variable dels paràmetres d'entrada de la funció mentre que el valor fora d'ells, és el tipus de variable que retorna. En aquest cas, la funció és molt senzilla i el seu paràmetre d'entrada és un *int* i retorna *void*, és a dir, res.

Com que l'objecte `StatusBar` és accessible des de la banda QML (veure apartat anterior), al fer:

```
property int primaryColor: Material.Blue

StatusBar {
    id: androidStatusBar
    color: Material.color(primaryColor, Material.Shade700)
}
```

S'assigna el color blau a l'*statusbar.*



Il·lustració 24: StatusBar de l'aplicació un cop se li ha definit el color

4.1.5. Peticions HTTP[39]

Al llarg de tota l'aplicació recorrem sovint el recurs de les peticions HTTP, ja sigui per fer una petició a la base de dades externa que gestiona els usuaris i els comandaments guardats, com per fer consultes a serveis de terceres parts (http://irdb.tk/)

Per a fer aquestes peticions s'ha creat un fitxer auxiliar en JavaScript anomenat XMLHttpRequestHelper.js. Aquest fitxer conté el següent codi:

```
function doXMLHttpRequest(method, host, params, callback, postHeader) {
    var url;
    var req;
    if(method === "GET")
    {
        url = host + "?" + params;
        console.log(url)
        req = new XMLHttpRequest();
        req.onreadystatechange =  //cada cop que l'estat cambia es crida la seguent funció
(stateChange)
        stateChange(req, callback);
        req.open('GET', url , true); //True = asyncronous
        req.send('');
    }
    else if (method === "POST")
    {
        url = host;
        req = new XMLHttpRequest();
        req.open('POST', url , true); //True = asyncronous
        if(postHeader)
            req.setRequestHeader("Content-type", postHeader);
        else
             req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        req.onreadystatechange = //cada cop que l'estat cambia es crida la seguent funció
(stateChange)
        stateChange(req, callback);
        req.send(params);
    }
    else if (method === "PUT")
    {
        url = host;
        req = new XMLHttpRequest();
        req.open('PUT', url , true); //True = asyncronous
        if(postHeader)
            req.setRequestHeader("Content-type", postHeader);
        else
             req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        req.onreadystatechange = //cada cop que l'estat cambia es crida la seguent funció
(stateChange)
        stateChange(req, callback);
        req.send(params);
    }
```

```
    else if (method === "PATCH")
    {
        url = host;
        req = new XMLHttpRequest();
        req.open('PATCH', url , true); //True = asyncronous
        if(postHeader)
            req.setRequestHeader("Content-type", postHeader);
        else
             req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        req.onreadystatechange = //cada cop que l'estat cambia es crida la seguent funció
(stateChange)
        stateChange(req, callback);
        req.send(params);
    }
    else if (method === "DELETE")
    {
        url = host;
        req = new XMLHttpRequest();
        req.open('DELETE', url , true); //True = asyncronous
        req.onreadystatechange = //cada cop que l'estat cambia es crida la seguent funció
(stateChange)
        stateChange(req, callback);
        req.send(null);
    }
}
function stateChange(myReq, callback) {
    return function() {
      console.log("Ready State " + myReq.readyState )
      console.log("Status: "  + myReq.status)
      if(myReq.readyState === XMLHttpRequest.DONE/*4*/ && myReq.status === 200)
      {
          callback(myReq);
      }
      else if(myReq.readyState === XMLHttpRequest.DONE/*4*/ && myReq.status !== 200)
          callback(myReq);
    }
}
```

Les dues funcions del fitxer permeten executar una petició HTTP asíncrona, que vol dir que un cop s'ha llençat la petició, el programa segueix corrent malgrat que la resposta del servidor encara no hagi arribat.

Gràcies a aquest fitxer auxiliar, quan volem fer una petició HTTP (GET, POST, PUT, PATCH, DELETE) en qualsevol lloc de l'aplicació simplement caldrà cridar la funció `doXMLHttpRequest` amb els paràmetres corresponents.

Com es pot veure alhora de definir els paràmetres que s'envien en la petició no hi ha diferència entre quina petició es desitja (GET, POST...) malgrat que després es processa de manera diferent a la funció `doXMLHttpRequest`, ja que en la funció GET els paràmetres es passen dintre la *url* mentre que en un POST/PATCH/PUT s'envien separadament com a paràmetres i en la DELETE no es tenen en compte.

Al llarg d'aquest apartat s'aniran veien diferents exemples de com es crida aquesta funció.

4.1.6. Configurar un comandament[3][4]

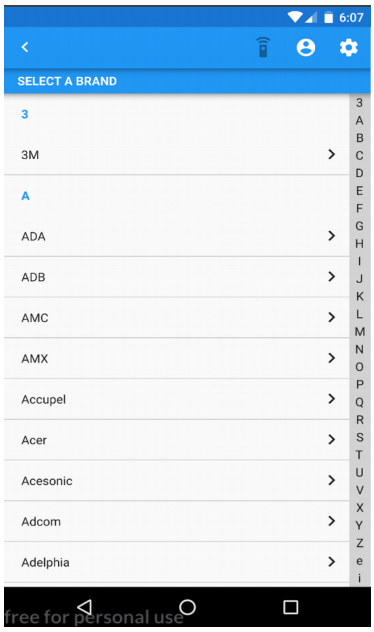
Una de les parts més importants i que han portat més mals de cap alhora de desenvolupar l'aplicació va ser trobar la manera d'obtenir els codis infrarojos que corresponen a cada botó de cada comandaments. Després de provar vàries opcions, la solució que més ens va agradar va ser http://irdb.tk/, una pàgina online que conté una gran base de dades de molts protocols infrarojos des d'on s'obtenen els codis. A més aquesta base de dades ofereix una API, que si ve és incompleta ja que no retorna els codis, és útil per anar poblant les diferents pantalles que l'usuari va passant fins a tenir el seu comandament configurat.

4.1.6.1. Marques disponibles

El primer pas per a configurar el comandament, és assignar quina marca és la desitjada. Al enviar la següent petició GET:

```
XMLHTTP.doXMLHttpRequest("GET", "http://irdb.tk/api/brand/","",function (rsp) { results(rsp)})
```

Aquesta ens retorna un JSON amb totes les marques de les quals la base de dades disposa de codis infrarojos. Aquest resposta ens permet poblar la següent llista:



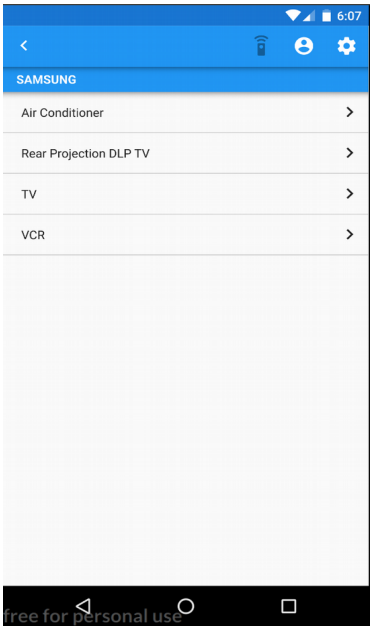
Il·lustració 25: Pantalla per a seleccionar la marca

4.1.6.2. Tipus de dispositius

El segon pas és definir quin tipus de dispositiu es vol controlar: TV, DVD, projector... L'API permet per una marca determinada demanar-li quins tipus de dispositius té disponibles. La petició que cal fer és:

```
XMLHTTP.doXMLHttpRequest("GET", "http://irdb.tk/api/devicetype/","brand=" + brand,function (rsp) {
results(rsp)})
```

On el nom de la marca (brand) és la seleccionada en l'anterior pantalla. De nou la resposta és un fitxer JSON que ens permet poblar la pantalla de seleccionar el tipus de dispositiu:

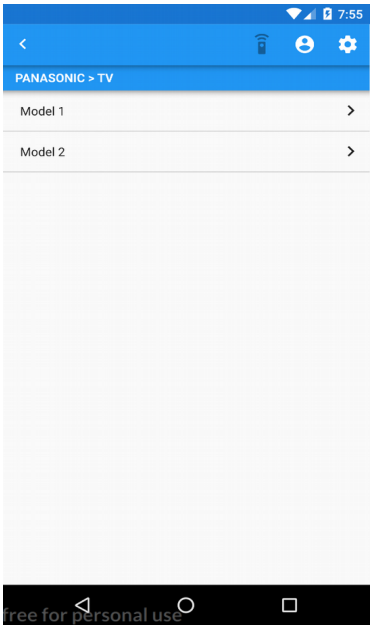


Il·lustració 26: Selecció del tipus de dispositiu

4.1.6.3. Model específic

L'últim pas que ha de configurar l'usuari és el de seleccionar el model específic, ja que per una mateixa marca i tipus de dispositiu pot ser que hi hagi dos protocols infraroig diferents. De nou l'API, té una petició especial per a tal de retornar-te quins models té disponibles:

```
XMLHTTP.doXMLHttpRequest("GET", "http://irdb.tk/api/codeset/","brand=" + brand + "&devicetype=" + deviceType,function (rsp) { results(rsp)})
```



Il·lustració 27: Selecció de model

4.1.6.4. Crear el comandament i assignar codi als botons

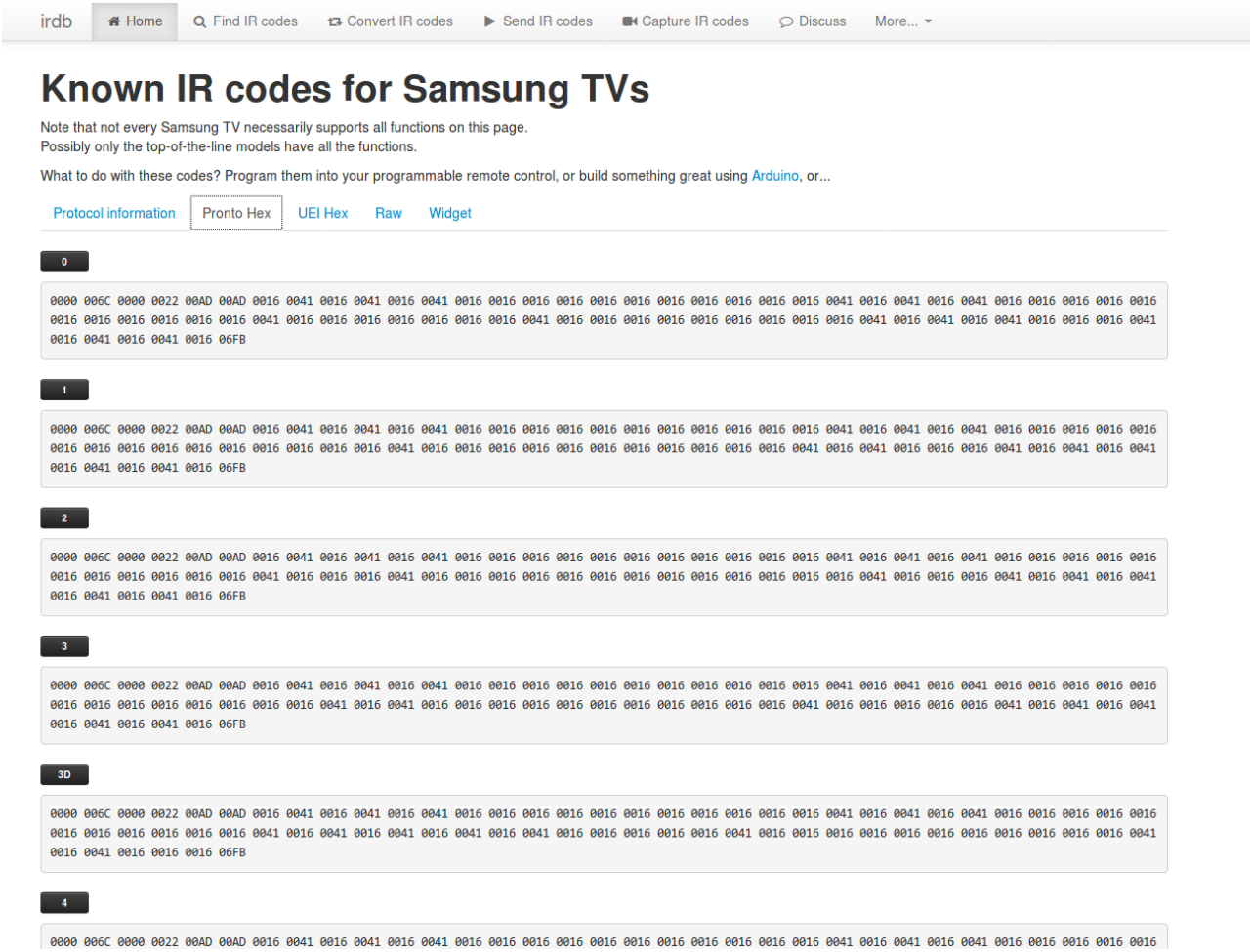
Sabent la marca, el tipus de dispositiu i el model, ja és suficient per tenir un comandament específic definit, ara bé, per mala sort, l'API no ofereix cap petició per tal de retornar-te els codis dels diferents botons del comandament. Simplement et retorna els diferents botons que el comandament conté sense més informació:

http://irdb.tk/api/code/?brand=JVC&devicetype=TV&protocol=JVC&device=3&subdevice=-1


```
{
  "meta": {
    "model": "code",
    "next": "",
    "page": 1,
    "previous": ""
  },
  "objects": [
    {
      "function": "",
      "protocol": "JVC",
      "subdevice": "-1",
      "devicetype": "TV",
      "device": "3",
      "functionname": "MAIN/SAP"
    },
    {
      "function": "3",
      "protocol": "JVC",
      "subdevice": "-1",
      "devicetype": "TV",
      "device": "3",
      "functionname": "SLEEP TIMER"
    },
    {
      "function": "4",
      "protocol": "JVC",
      "subdevice": "-1",
      "devicetype": "TV",
      "device": "3",
      "functionname": "DISPLAY"
    },
(...)
```


Per tant, via API no hi ha manera d'obtenir els codis de cada botó (tal i com està explicat al capítol 2.3.1).

Mitjançant la interfície web[3], però, si que es poden aconseguir els codis:



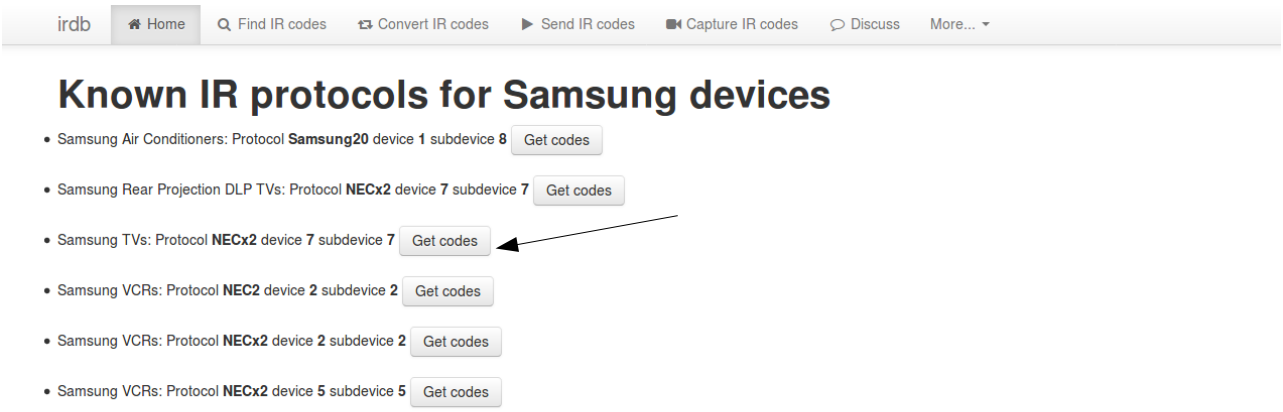
Il·lustració 28: Codis de les trames infraroges dels diferents botons del comandament Samsung per a TV

Ara bé, aquest procés l'hem d'aconseguir automatitzar per tal que l'aplicació ho pugui fer.

Com que el mètode per aconseguir aquests codis és bastant complex, s'abordarà amb més detall al següent apartat.

4.1.7. Obtenció dels codis del protocol infraroig

Com anteriorment s'ha comentat, és impossible aconseguir els codis infrarojos mitjançant l'API de la base de dades. És per això que hem de replicar la petició que fa la web al polsar sobre *GET CODES*:

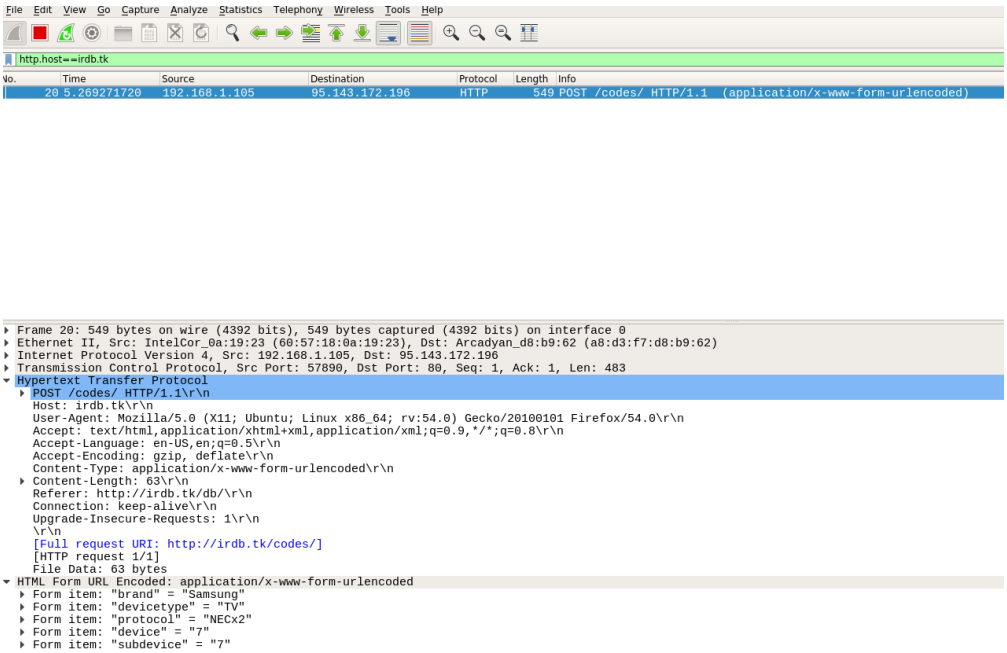


Il·lustració 29: Interfície de la web just abans d'aconseguir els codis del protocol infraroig

Malauradament, la petició que li fa a la base de dades és una petició POST, per tant els paràmetres no es veuen a la *url.* El procés de replicar la petició es fa més complicat però no impossible.

Gràcies a un analitzador de tràfic a la xarxa com pot ser el WireShark[6], es poden capturar les peticions POST amb els seus paràmetres.

El que es fa és obrir el WireShark i filtrar només el tràfic que passa per la *url* que conté http://irdb.tk/, ja que sinó hi hauria massa tràfic i es faria complicat localitzar la petició. Un cop configurat el filtre, es polsa sobre el botó *GET CODES* i al WireShark apareix:



Il·lustració 30: Captura de pantalla del WireShark just després de fer la petició POST des de la web de http://irdb.tk/

A la imatge es veu la petició POST amb tots els seus paràmetres, amb més detall:

```
    [Full request URI: http://irdb.tk/codes/]
    [HTTP request 1/1]
    File Data: 63 bytes
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "brand" = "Samsung"
  ▸ Form item: "devicetype" = "TV"
  ▸ Form item: "protocol" = "NECx2"
  ▸ Form item: "device" = "7"
  ▸ Form item: "subdevice" = "7"
```

Il·lustració 31: Detall de la petició POST

Per tant gràcies a la informació que s'obté del WireShark ja som capaços de replicar la petició post:

```
XMLHTTP.doXMLHttpRequest("POST", "http://irdb.tk/codes","brand=" + brand +"&devicetype=" + deviceType + "&protocol=" + protocol + "&device=" + device + "&subdevice=" + subdevice,function (rsp) {parseHtml(rsp)})
```

Aquesta petició, no retorna un fitxer JSON o XML fàcilment parsejable, sinó que directament retorna l'HTML mateix que l'explorador web (Chrome, Firefox…) processa per a mostrar la pàgina (imatge 28.) Per tant un s'obté la resposta com una string de tot l'HTML cal parsejar-lo per obtenir els codis:

```
<a class="tip" data-toggle="tooltip" title="Original Button Code, also known as Function">OBC</a> 11</span></td>
</tr>

</table>
    </div>
    <div class="tab-pane" id="pronto">

    <p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">0</span></button></p>
    <pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004

    <p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">1</span></button></p>
    <pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004

    <p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">2</span></button></p>
    <pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004

    <p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">3</span></button></p>
    <pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004

    <p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">3D</span></button></p>
    <pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004

    <p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">4</span></button></p>
    <pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004
```

Il·lustració 32: Fragment del HTML que interessa parsejar

De tot l'HTML que conté més de 800 línies només volem obtenir el nom del botons i el seu codi del protocol infraroig. Per això cridem la funció *parseHTML()*.

```
function parseHtml(rsp)
{
    console.log(rsp.responseText)
    var usefulText = rsp.responseText;
    usefulText = usefulText.substr(usefulText.indexOf('<div class="tab-pane" id="pronto">'));
    usefulText = usefulText.substr(0, usefulText.indexOf('<div class="tab-pane" id="hexa">'));
    usefulText = usefulText.substr(usefulText.indexOf('<p>'));
    usefulText = usefulText.substr(0, usefulText.indexOf('</div>'));
    var buttonsInfoFromHtml = usefulText.split("</pre>");
    buttonsInfoFromHtml.pop();
    for(var i in buttonsInfoFromHtml)
    {
        parseButtonFromHtml(buttonsInfoFromHtml[i]);
    }
    isReady(buttons)
}
```

Aquesta funció crea una variable *buttonsInfoFromHtml* que es un array amb tots la informació dels els botons en format HTML.

Tot seguit el que es fa és per a cada botó HTML, es parseja per obtenir-ne el seu nom i el seu codi mitjançant la funció *parseButtonFromHTML.*

```
function parseButtonFromHtml(buttonInfoFromHtml)
{
    var button = {
        name: "",
        prontoCode: ""
    }
    button.name = buttonInfoFromHtml.substr(0,buttonInfoFromHtml.indexOf('</span>'));
    button.name = button.name.substr(button.name.indexOf('300px">') + '300px">'.length );
    button.prontoCode = buttonInfoFromHtml.substr(buttonInfoFromHtml.indexOf("prettyprinted'>") +
"prettyprinted'>".length);
    buttons.push(button);
}
```

Tots aquests botons s'emmagatzemen a una variable anomenada *buttons* que també és una array amb tots els botons.

```
▼ buttons
  ▼ 0
      name        "0"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 1
      name        "1"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 10
      name        "9"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 11
      name        "AD/SUBT"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 12
      name        "CH LIST"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 13
      name        "CHANNEL +"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 14
      name        "CHANNEL -"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 15
      name        "CURSOR DOWN"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
  ▼ 16
      name        "CURSOR LEFT"
      prontoCode  "0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 001
```

Il·lustració 33: Variable *buttons* un cop ha estat poblada

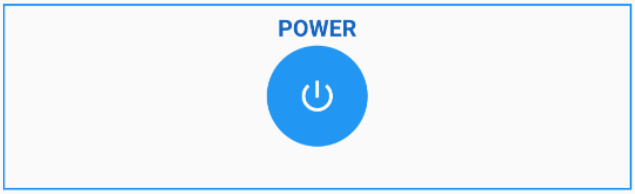
Amb la variable *buttons* poblada tan sols fa falta assignar el contingut d'aquests botons als *widgets* del comandament. Com a *widget* ens referim als elements visuals que formen al comandament:

Il·lustració 34: Exemple de Widget

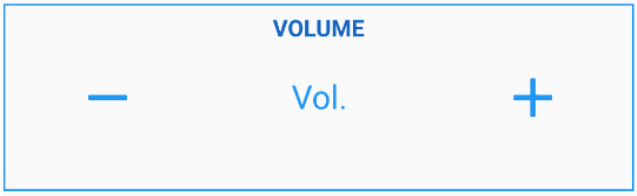
4.1.8. Creació dels Widgets

Els widgets, tal i com s'ha comentat a l'anterior apartat són el conjunt d'elements visuals que configuren el *layout* del comandament.

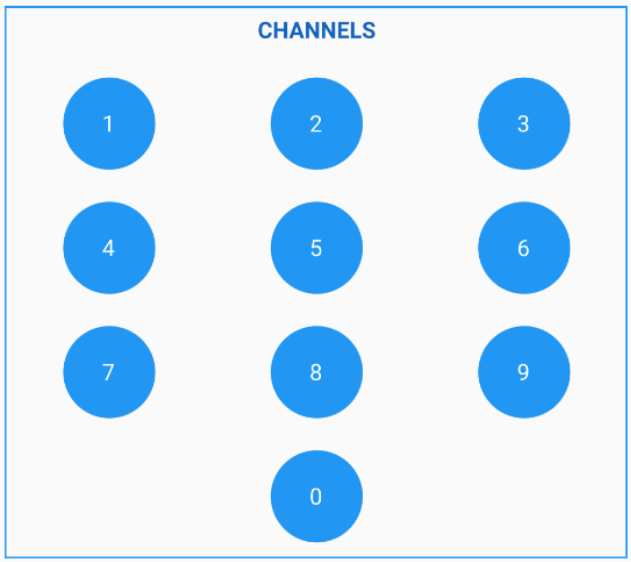
Hi ha widgets que només tenen un botó associat, però n'hi ha que en tenen 2 i fins i tot més.



Il·lustració 35: Widget amb 1 botó associat



Il·lustració 36: Widget amb 2 botons associats



Il·lustració 37: Widget amb múltiples botons associats

La funció que s'encarrega de buscar a la variable `buttons` i en funció dels seus valors crear els widgets assignant-los-hi els botons és una mica rebuscada tal com es mostra a continuació:

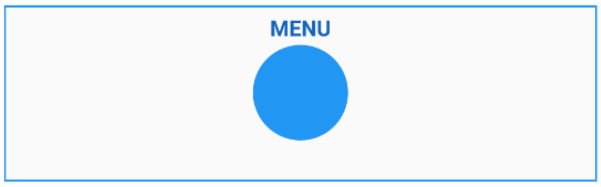
```
function createAndDrawWidgets()
{
    function Widget (){
        this.name = "";
        this.prontoCode = []
    }
    function WidgetIndex(){
        this.index = 0;
    }
    function isInArray(value, array, widgetIndex) {
      widgetIndex.index = array.findIndex(function(element){return element.name === value});
      return array.findIndex(function(element){return element.name === value}) > -1;
    }
    for(var i in buttons)
        auxButtons.push(buttons[i]);
    var w = new Widget();
    var i = new WidgetIndex();
    if(isInArray("POWER", buttons, i))
    {
        w.name = "POWER";
        w.prontoCode.push(buttons[i.index].prontoCode);
        buttons.splice(i.index, 1);
        widgets.push(w);
    }
    w = new Widget();
    i = new WidgetIndex();
    if(isInArray("VOLUME +", buttons, i))
    {
        w.name = "VOLUME";
        w.prontoCode.push(buttons[i.index].prontoCode);
        buttons.splice(i.index, 1);
        i = new WidgetIndex();
        if(isInArray("VOLUME -", buttons, i))
        {
            w.prontoCode.push(buttons[i.index].prontoCode)
            buttons.splice(i.index, 1);
        }
        widgets.push(w);
    }
    w = new Widget();
    i = new WidgetIndex();
    if(isInArray("CHANNEL +", buttons, i))
    {
        w.name = "CHANNEL";
        w.prontoCode.push(buttons[i.index].prontoCode);
        buttons.splice(i.index, 1);
        i = new WidgetIndex();
        if(isInArray("CHANNEL -", buttons, i))
        {
            w.prontoCode.push(buttons[i.index].prontoCode)
            buttons.splice(i.index, 1);
        }
        widgets.push(w);
    }
    w = new Widget();
    i = new WidgetIndex();
    if(isInArray("0", buttons, i))
    {
        w.name = "CHANNELS";
        w.prontoCode.push(buttons[i.index].prontoCode);
        buttons.splice(i.index, 1);
        for(var n = 1; n<=9; n++)
        {
            i = new WidgetIndex();
            if(isInArray(n.toString(), buttons, i))
```

```
            {
                w.prontoCode.push(buttons[i.index].prontoCode)
                buttons.splice(i.index, 1);
            }

<p><button class="btn btn-inverse btn-mini" style="min-width: 50px"><span style="font-weight: bold; width: 300px">4</span></button></p>
<pre class='prettyprint prettyprinted'>0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 004

            else
            {
               w.prontoCode.push("");
            }
        }
        widgets.push(w);
    }
    for (var i in buttons)
    {
        var w = new Widget;
        w.name = buttons[i].name;
        w.prontoCode.push(buttons[i].prontoCode);
        buttons.splice(i, 1);
        if(w.name !== "")
            widgets.push(w);
    }
     for (var i in widgets)
    {
        if(widgets[i].name === "POWER")
        {
            powerWidget.name = widgets[i].name;
            powerWidget.prontoCode = widgets[i].prontoCode[0];
            widgetsHolder.children.push(powerWidget);
        }
        else if(widgets[i].name === "VOLUME")
        {
            volumeWidget.name = widgets[i].name;
            volumeWidget.prontoCodeUp = widgets[i].prontoCode[0];
            volumeWidget.prontoCodeDown = widgets[i].prontoCode[1];
            widgetsHolder.children.push(volumeWidget);
        }
        else if(widgets[i].name === "CHANNEL")
        {
            channelWidget.name = widgets[i].name;
            channelWidget.prontoCodeUp = widgets[i].prontoCode[0];
            channelWidget.prontoCodeDown = widgets[i].prontoCode[1];
            widgetsHolder.children.push(channelWidget);
        }
        else if(widgets[i].name === "CHANNELS")
        {
            channelNumbersWidget.name = widgets[i].name;
            channelNumbersWidget.prontoCode0 = widgets[i].prontoCode[0];
            channelNumbersWidget.prontoCode1 = widgets[i].prontoCode[1];
            channelNumbersWidget.prontoCode2 = widgets[i].prontoCode[2];
            channelNumbersWidget.prontoCode3 = widgets[i].prontoCode[3];
            channelNumbersWidget.prontoCode4 = widgets[i].prontoCode[4];
            channelNumbersWidget.prontoCode5 = widgets[i].prontoCode[5];
            channelNumbersWidget.prontoCode6 = widgets[i].prontoCode[6];
            channelNumbersWidget.prontoCode7 = widgets[i].prontoCode[7];
            channelNumbersWidget.prontoCode8 = widgets[i].prontoCode[8];
            channelNumbersWidget.prontoCode9 = widgets[i].prontoCode[9];
            channelNumbersWidget.enableNum0 = widgets[i].prontoCode[0] !== "";
            channelNumbersWidget.enableNum1 = widgets[i].prontoCode[1] !== "";
            channelNumbersWidget.enableNum2 = widgets[i].prontoCode[2] !== "";
            channelNumbersWidget.enableNum3 = widgets[i].prontoCode[3] !== "";
            channelNumbersWidget.enableNum4 = widgets[i].prontoCode[4] !== "";
            channelNumbersWidget.enableNum5 = widgets[i].prontoCode[5] !== "";
            channelNumbersWidget.enableNum6 = widgets[i].prontoCode[6] !== "";
```

```
            channelNumbersWidget.enableNum7 = widgets[i].prontoCode[7] !== "";
            channelNumbersWidget.enableNum8 = widgets[i].prontoCode[8] !== "";
            channelNumbersWidget.enableNum9 = widgets[i].prontoCode[9] !== "";
            widgetsHolder.children.push(channelNumbersWidget);
        }
        else
        {
            var widgetTemp = Qt.createComponent("./RemoteWidgets/RoundButtonWidget.qml");
            if (widgetTemp.status === Component.Ready)
                widgetsHolder.children.push( widgetTemp.createObject(container, {
                width: Qt.binding(function() { return widgetsHolder.cellWidth }),
                height: Qt.binding(function() { return widgetsHolder.cellHeight }),
                name: widgets[i].name,
                prontoCode: widgets[i].prontoCode[0]}));
        }
    }
    for(var i in auxButtons)
        buttons.push(auxButtons[i])
    isLoading = false;
}
```

Aquesta funció permet crear la interfície de cada comandament de manera dinàmica; és la pròpia funció la que va col·locant els controls a la pantalla mentre va processant la resposta de la base de dades de irdb.

A trets generals aquesta funció consulta a tot l'array *buttons* que conté tots els botons que tenim del comandament. Posteriorment, pels widgets personalitzats: POWER, VOLUME, CHANNEL +/-, CHANNELS va a buscar els botons que corresponent a cada element del widget i li carrega els codis infrarojos. Per la resta de widgets que són genèrics, simplement es crea un widget estàndard amb un botó i un títol i se li carrega el codi també.



Il·lustració 38: Exemple de widget genèric

4.1.9. PRONTO CODES => RAW CODES

Els codis que s'obtenen de la base de dades http://irdb.tk/, ens arriben en un format anomenat prontoCode[40] , aquest format cal convertir-lo en el format anomenat *RAW CODE* per tal de poder enviar els polsos en infraroig, ja que el format RAW CODE no és res més que una seqüència de polsos on alternadament hi ha el valor en microsegons d'un pols en estat alt i d'un pols en estat baix:

```
{+9024 -4512 +564 -564 +564 -564 +564 -564 +564 -564 +564 -564 +564 -564 +564 -564 +564 -564 +564 -1692
+564 -1692 +564 -1692 +564 -1692 +564 -1692 +564 -1692 +564 -564 +564 -1692 +564 -564 +564 -1692 +564
-564 +564 -564 +564 -564 +564 -564 +564 -564 +564 -564 +564 -1692 +564 -564 +564 -1692 +564 -1692 +564
-1692 +564 -1692 +564 -1692 +564 -1692 +564 -40884}
```

En canvi, el format prontoCode té aquest aspecte

```
0000 006C 0022 0002 015B 00AD 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016
0016 0016 0041 0016 0041 0016 0041 0016 0041 0016 0041 0016 0041 0016 0016 0016 0041 0016 0016 0016 0041
0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 0016 0041 0016 0041 0016
0041 0016 0041 0016 0041 0016 0041 0016 0622 015B 0057 0016 0E6C
```

On:

- El 1r paquet sempre val 0.
- El segon paquet és la freqüència de la portadora de la senyal IR. Assumint que el valor del segon paquet és n:

$$Freq[Hz] = \frac{1000000}{n \cdot 0.241246}$$

- El tercer i el quart paquet fan referència al nombre de polsos que tindrà la trama
- La resta de paquets vindria a ser l'equivalent al *RAW CODE* codificats de la següent manera. Assumint *n* com el valor del paquet i *f* com la freqüència obtinguda del 2n paquet.

$$PulseTime[us] = n \cdot \frac{1}{f[Hz]} \cdot 1000000$$

Tot seguit es mostra la funció que entrant el prontoCode retorna el Raw Code i la freqüència de la portadora:

```
function decodeProntoHexCode(prontoHexCode_String)
{
    var rsp = {
        carrierFreq: 0,
        rawCode: []
    };
    var prontoCodeValues = [];
    var rawCode = [];
    var carrierFreq;
    var periodOfCarrier;
    var aux = prontoHexCode_String.split(" ");
    for (var i in aux)
    {
        prontoCodeValues.push(parseInt(aux[i], 16));
    }
    if(!prontoCodeValues.shift())
        carrierFreq = prontoCodeValues.shift();
    carrierFreq = (1000000/(carrierFreq * 0.241246))
    periodOfCarrier = (1/carrierFreq) * 1000000;
    prontoCodeValues.shift();
    prontoCodeValues.shift()
    for (i in prontoCodeValues)
    {
        rawCode.push(Math.round(prontoCodeValues[i] * periodOfCarrier));
    }
    rsp.carrierFreq = Math.round(carrierFreq);
    rsp.rawCode = rawCode
    return rsp;
}
```

4.1.10. Enviament de trames infraroges

Finalment, després de tots els passos i obtenir els codis (en format *Raw*) del protocol infraroig, cal que el port infraroig els enviï realment. Aquesta funcionalitat no està integrada en Qt (ni C++ ni QML) i per tant haurem d'escriure una classe en Java.

En Java, existeix una classe anomenada ConsumerIrManager[41] que permet enviar trames infraroges mitjançant l'emissor que alguns dispositius Android porten incorporat.

El nostre codi Java que fa aquesta gestió és el següent:

IrManager.java:

```
package cat.selrac;
import android.hardware.ConsumerIrManager;
import android.content.Context;
public class IrManager extends MyActivity
{
    private static ConsumerIrManager mCIR;
    public IrManager()
    {
    }
    public static boolean checkIfHasEmmiter(int c)
    {
        if (mCIR == null) {
            mCIR = (ConsumerIrManager)qtActivity.getSystemService(Context.CONSUMER_IR_SERVICE);
        }
        return mCIR.hasIrEmitter();
    }
    public static void sendIrFrame(int carrierFrequency,int[] pattern)
    {
        if( mCIR.hasIrEmitter())
        {
            mCIR.transmit(carrierFrequency, pattern);
        }
    }
}
```

Aquest classe implementa dues funcions: la primera anomenada checkIfHasEmmiter per tal de comprovar si el dispositiu mòbil té o no el transmissor infraroig, i la segona anomenada sendIrFrame per a enviar la trama infraroja.

Per accedir a aquesta classe Java ha sigut necessari crear una classe en Qt C++:

IrManager.h:

```
#ifndef IRMANAGER_H
#define IRMANAGER_H
#include <QObject>
#ifdef Q_OS_ANDROID
#include <QtAndroid>
#include <QAndroidJniEnvironment>
#endif
class IrManager : public QObject
{
    Q_OBJECT
    Q_PROPERTY(bool available READ isAvailable CONSTANT)
public:
    explicit IrManager(QObject *parent = nullptr);
    Q_INVOKABLE void sendIrFrame(int carrierFrequency, QList<int> pattern);
    bool isAvailable() const;
private:
};
#endif // STATUSBAR_H
```

StatusBar.cpp:

```
#include "irmanager.h"
IrManager::IrManager(QObject *parent) : QObject(parent)
{
}
bool IrManager::isAvailable() const
{
#ifdef Q_OS_ANDROID
    if(QtAndroid::androidSdkVersion() >= 19)
    {
        return QAndroidJniObject::callStaticMethod<jboolean>("cat/selrac/IrManager",
                                                             "checkIfHasEmmiter",
                                                             "(I)Z",
                                                              0);
    }
    else
        return false;
#else
    return false;
#endif
}
void IrManager::sendIrFrame(int carrierFrequency, QList<int> pattern )
{
    if (!isAvailable())
        return;
    int patternArray[500];
    for(int i=0; i<pattern.length(); i++)
        patternArray[i] = pattern[i];
#ifdef Q_OS_ANDROID
    QtAndroid::runOnAndroidThread([=]() {
        QAndroidJniEnvironment env;
        jintArray patternArray_jin = env->NewIntArray(pattern.length());
        env->SetIntArrayRegion(patternArray_jin, 0, pattern.length(), patternArray);
        QAndroidJniObject::callStaticMethod<void>("cat/selrac/IrManager",
                                                  "sendIrFrame",
                                                  "(I[I)V",
                                                   carrierFrequency,
                                                   patternArray_jin);
     });
#endif
}
```

En la funció de C++, sendIrFrame, el codi Raw és un array de ints. Com a curiositat cal comentar que ha hagut de "patir" dos canvis de tipatge per tal de poder ser enviada cap a Java, ja que la funció demanava que el paràmetre fos un `jintArray` (que per assignar-li els valors es pot fer partint d'un int array però no d'un QList<int> que és el valor d'entrada de la funció ja que des de QML no es pot enviar un int array).

Finalment des de la banda QML, quan volem enviar una trama cal fer:

```
if(!IrManager.available)
    console.log("IR Blaster is not available")
else
    IrManager.sendIrFrame(carrierFrequency, rawCode);
```

Primer es mira si el dispositiu té el emissor infraroig disponible i en cas afirmatiu es crida la funció *sendIrFrame* definida a *IrManager.cpp*.

4.1.11. Gestió d'usuaris

Per a la gestió d'usuaris es crea un objecte QML amb tot un seguit de propietats i funcions, aquest objecte l'anomenem User i s'encarregarà de gestionar tots els processos que tenen relació amb la gestió d'usuaris. Aquest objecte s'instancia a la variable *user* al fitxer main.qml, per tal que sigui accessible des de tot el codi QML.

```
property User user: User{}
```

La gestió d'usuaris en si, com està explicat al capítol 3.2.3.2 es fa des dels serveis de *Firebase*. Per tant , l'objecte *user* simplement anirà interaccionant amb l'API de Firebase.

L'objecte User, té un seguit de propietats que s'aniran modificant al cridar-se les diferents funcions. Aquestes propietats són:

```
property string idToken
property string localId
property bool isLogged: false
property string userName
property string password
property string email
property bool doingThings: false
```

Al fet de crear aquest objecte i assignar les funcions dintre seu, permet que posteriorment quan cal fer ús d'aquestes en el codi, la manera de cridar-les és molt neta i senzilla, per exemple:

```
user.signUp(textFieldEmail.text, textFieldPassword.text, textFieldUserName.text, checkBoxKeepLogged.checked);
```

4.1.11.1. Nou Registre

Alhora de registrar un nou usuari es crida la següent funció:

```
function signUp(email, password, username, keepLogged){
    doingThings = true;
    XMLHTTP.doXMLHttpRequest("POST",
"https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?
key=AIzaSyBKZ9_378TMs5es8sIG9ZTbFTf5_aMBp0o",
                             '{"email":"'+ email + '","password":"'+ password + '","displayName":"'+
username + '","returnSecureToken":true}', function (rsp) { parse(rsp)}, "application/json")
    function parse(rsp)
    {
        console.log(rsp.responseText);
        if(rsp.status === 0)
        {
            doingThings = false;
            alert.text = qsTr("There is no connexion");
            alert.open();
            return;
        }
        var res = JSON.parse(rsp.responseText);
        if(res.hasOwnProperty("error")){
            alert.text = (  qsTr("Error code: ") + res.error.code + qsTr("\nError: ") +
res.error.message + Translator.emptyString);
            alert.open();
```

```
            doingThings = false;
            return;
        }
        idToken = res.idToken;
        localId= res.localId;
        userName = username;
        user.password = password;
        user.email = email;
        createAccountOnDatabase();
    }
    function createAccountOnDatabase()
    {
         XMLHTTP.doXMLHttpRequest("PUT", "https://allinoneirremote.firebaseio.com/users/" + localId +
".json?auth=" + idToken,
                                  '{"email":"'+ email + '","username":"'+ username + '","password":"'
+ password +'","myCloudRemotes":"[]"}', function (rsp) { parse(rsp)}, "application/json")
        function parse(rsp)
        {
            console.log(rsp.responseText);
            if(keepLogged)
            {
                db.dbUpdate("email", email);
                db.dbUpdate("password", password);
            }
            isLogged = true;
            doingThings = false;
        }
    }
}
```

Aquesta funció fa una petició a *Firebase* amb el correu i la contrasenya de l'usuari que es vol registrar. *Firebase* retorna sempre una resposta en format JSON. Si aquesta resposta indica que hi hagut un error, es surt de la funció i es mostra l'error a la pantalla.

Si la petició a sigut satisfactòria s'omplen les propietats de l'usuari amb els valors retornats per a la petició i es crea l'usuari també a una base de dades de *Firebase* que és on posteriorment guardarem els comandaments de l'usuari.

A més en el cas que l'usuari vulgui mantenir la sessió al següent cop que obri l'aplicació, les credencials de l'usuari (e-mail i contrasenya) es guarden a la base de dades locals.

4.1.11.2. Inici de sessió

Per tal de fer un inici de sessió o LogIn, la funció és força similar que alhora de registrar-se, fent una petició diferent a *Firebase* i sense crear l'usuari a la base de dades externa ja que ja existeix.

```
function logIn(email, password, keepLogged){
    doingThings = true;
    XMLHTTP.doXMLHttpRequest("POST",
"https://www.googleapis.com/identitytoolkit/v3/relyingparty/verifyPassword?
key=AIzaSyBKZ9_378TMs5es8sIG9ZTbFTf5_aMBp0o",
                             '{"email":"'+ email + '","password":"'+ password +
'","returnSecureToken":true}', function (rsp) { parse(rsp)}, "application/json")
```

```
function parse(rsp)
    {
        console.log(rsp.responseText);
        if(rsp.status === 0)
        {
            doingThings = false;
            alert.text = qsTr("There is no connexion");
            alert.open();
            return;
        }
        var res = JSON.parse(rsp.responseText);
        if(res.hasOwnProperty("error")){
            alert.text = (  qsTr("Error code: ") + res.error.code + qsTr("\nError: ") +
res.error.message + Translator.emptyString);
            alert.open();
            logOut();
            doingThings = false;
            return;
        }
        idToken = res.idToken;
        localId= res.localId;
        userName = res.displayName;
        user.password = password;
        user.email = email;
        doingThings = false;
        if(keepLogged)
        {
            db.dbUpdate("email", email);
            db.dbUpdate("password", password);
        }
        isLogged = true;
    }
}
```

4.1.11.3. Tancar sessió

Quan l'usuari decideix tancar la sessió, el procés que es segueix és molt senzill. Simplement es canvien els valors de les propietats de User i s'assegura que a la base de dades s'esborrin les credencials.

```
function logOut(){
    doingThings = true;
    idToken = "";
    localId = "";
    userName = "";
    user.password = "";
    user.email = "";
    db.dbUpdate("email", "");
    db.dbUpdate("password", "");
    isLogged = false;
    doingThings = false;
}
```

4.1.11.4. Eliminar usuari

Quan l'usuari es vol donar de baixa, crida la següent funció:

```
XMLHTTP.doXMLHttpRequest("POST",
"https://www.googleapis.com/identitytoolkit/v3/relyingparty/deleteAccount?
key=AIzaSyBKZ9_378TMs5es8sIG9ZTbFTf5_aMBp0o",
                        '{"idToken":"'+ idToken + '"}', function (rsp) { parse(rsp)},
"application/json")
function parse(rsp)
{
    console.log(rsp.responseText);
    if(rsp.status === 0)
    {
        doingThings = false;
        alert.text = qsTr("There is no connexion");
        alert.open();
        return;
    }
    var res = JSON.parse(rsp.responseText);
    if(res.hasOwnProperty("error")){
        alert.text = (  qsTr("Error code: ") + res.error.code + qsTr("\nError: ") + res.error.message
+ Translator.emptyString);
        alert.open();
        doingThings = false;
        return;
    }
    deleteAccountOnDataBase();
}
function deleteAccountOnDataBase()
{
    XMLHTTP.doXMLHttpRequest("DELETE", "https://allinoneirremote.firebaseio.com/users/" + localId +
".json?auth=" + idToken,
                             '{}', function (rsp) { parse(rsp)}, "X-HTTP-Method-Override: DELETE")
   function parse(rsp)
   {
       console.log(rsp.responseText);
       logOut();
   }
}
}
```

Després de fer la petició corresponent a *Firebase,* es fa una gestió del retorn i si no hi hagut cap error s'elimina també l'usuari a la base de dades externa i per acabar es força un logout de l'usuari per tal de fer un *clear* a les propietats de l'objecte User.

4.1.11.5. Sobreescriure els comandaments associats a l'usuari emmagatzemats a la base de dades externa

Quan es volen pujar els comandaments emmagatzemats en local a la base de dades externa es crida:

```
function updateMyCloudRemotes(myCloudRemotesInJson)
{
    XMLHTTP.doXMLHttpRequest("PATCH", "https://allinoneirremote.firebaseio.com/users/" + localId +
".json?auth=" + idToken,
                             '{"myCloudRemotes":'+ myCloudRemotesInJson +'}', function (rsp) {
parse(rsp)}, "application/json")
   function parse(rsp)
   {
       console.log(rsp.responseText);
       updateMyCloudRemotesFinished();
   }
}
```

Un paràmetre molt importat d'aquesta funció és *localId.* Ja que el seu valor s'obté al iniciar la sessió i és únic per a cada usuari. De tal manera que quan ataquem a la base de dades externa, gràcies a aquest paràmetre estem atacant als comandaments relacionats amb l'usuari que fa la petició.

4.1.11.6. Obtenir els comandaments associat a l'usuari de la base de dades externa

Per tal de dur a terme aquesta tasca, es crida:

```
function getMyCloudRemotes()
{
    XMLHTTP.doXMLHttpRequest("GET", "https://allinoneirremote.firebaseio.com/users/" + localId +
".json?auth=" + idToken,
                             "", function (rsp) { parse(rsp)}, "application/json")
   function parse(rsp)
   {
       var dataBaseContent = JSON.parse(rsp.responseText);
       var myCloudRemotes = dataBaseContent.myCloudRemotes;
       getMyCloudRemotesFinished(myCloudRemotes);
   }
}
```

Un paràmetre molt importat d'aquesta funció és *localId.* Ja que el seu valor s'obté al iniciar la sessió i és únic per a cada usuari. De tal manera que quan ataquem a la base de dades externa, gràcies a aquest paràmetre estem obtenint els comandaments relacionats amb l'usuari que fa la petició.

4.1.12. Base de dades local[42]

En l'aplicació del present projecte s'implementa una base de dades local per satisfer bàsicament dos necessitats:

1. Permetre a l'usuari mantenir-se connectat al reobrir l'aplicació.

2. Fer que l'aplicació pugui emmagatzemar comandaments per tal que l'aplicació sigui operativa malgrat no tinguem connexió a internet.

Com en el cas de la gestió d'usuari, per a gestionar la base de dades es crea un objecte QML anomenat DataBase que implementa totes les funcions necessàries pera interactuar amb la base de dades interna, que es tracta d'una base de dades SQLite.

Per a poder interactuar cal importar:

```
import QtQuick.LocalStorage 2.0 as Sql
```

Les diferents funcions que implementa són:

4.1.12.1. dbInit

```
function dbInit()
{
    var db = Sql.LocalStorage.openDatabaseSync("AllInOneIrRemote_DB", "", "All Local Storage Here",
1000000);
    try {
        db.transaction(function (tx) {
            tx.executeSql('CREATE TABLE IF NOT EXISTS localStorage (key text unique,value text)')
        })
    } catch (err) {
        console.log("Error creating table in database: " + err)
    };
    dbInsert("email", "");
    dbInsert("password", "");
    dbInsert("myRemotes", "[]");
}
```

Obre i crea una base de dades i després en el cas que no existeixi crea una taula anomenada *localStorage* on s'emmagatzemaran les dades. Com està explicat a l'apartat 3.2.3.1 cada fila de la taula només tindrà una *key* i un *value*.

El primer cop que es cridi aquesta funció, a més afegirà una fila per a l'e-mail de l'usuari, una altra per la contrasenya i una altra per als comandaments. A totes aquestes files se'ls hi assigna (columna `value`) un valor nul.

4.1.12.2. dbGetHandle

```
function dbGetHandle()
{
    try {
        var db = Sql.LocalStorage.openDatabaseSync("AllInOneIrRemote_DB", "", "All Local Storage Here",
1000000);
    } catch (err) {
        console.log("Error opening database: " + err)
    }
    return db
}
```

Aquesta funció la criden totes les altres per a tal de saber cap a quina base de dades han de fer l'acció.

4.1.12.3. dbUpdate

```
function dbUpdate(pKey, pValue)
{
    var db = dbGetHandle()
    try{
        db.transaction(function (tx) {
            tx.executeSql(
                        'update localStorage set value=? where key = ?', [pValue, pKey])
        })
    }catch(err){
        console.log("Error updating " + pKey + ": " + err)
    };
}
```

Aquesta funció canvia el valor de *value* de la fila desitjada.

4.1.12.4. dbInsert

```
function dbInsert(pKey, pValue)
{
    var db = dbGetHandle()
    var rowid = 0;
    try{
        db.transaction(function (tx) {
            tx.executeSql('INSERT INTO localStorage VALUES(?, ?)',
                          [pKey, pValue])
            var result = tx.executeSql('SELECT last_insert_rowid()')
            rowid = result.insertId
        })
    }catch(err){
        console.log("Error inserting " + pKey + ": " + err)
    };
    return rowid;
}
```

Aquesta funció inserta una fila a la base de dades, només s'utilitza el primer cop que es crida `dbInit`();

4.1.12.5. dbReadAll

```
function dbReadAll()
{
    var rsp = {};
    var db = dbGetHandle()
    try{
        db.transaction(function (tx) {
            var results = tx.executeSql(
                        'SELECT * FROM localStorage')
            for (var i = 0; i < results.rows.length; i++) {
                rsp[results.rows.item(i).key] = results.rows.item(i).value;
            }
        })
    }catch(err){
        console.log("Error reading table: " + err)
    };
    return rsp;
}
```

Aquesta funció retorna totes les dades que té la base de dades en forma d'objecte JavaScript, fet que és senzill de realitzar gràcies a que la configuració de la taula tan sols té les columnes *key* i *value.*

4.2. Part del servidor[17][18]

El servidor que fem servir per a gestionar els usuaris o per crear la base de dades externa, és un servei de Google anomenat *Firebase.* Per tant no ha calgut programar el servidor, però si crear-lo i configurar-lo.

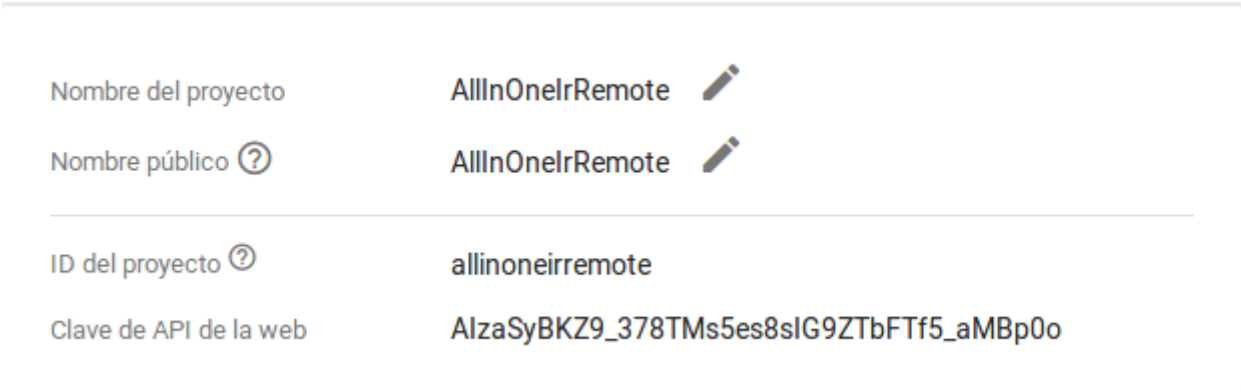
En aquest apartat s'explica breument com s'ha configurat el servidor.

Per tal de tenir un servidor propi, cal crear-lo. Aquest procés es tan fàcil com entrar al teu compte de google i polsar sobre *Crear un nou projecte*. Apareix un PopUp i s'assigna un nom al projecte.

4.2.1. Firebase Authentication[25]

Una de les funcionalitats que s'utilitza és la *Firebase Authentication,* que s'encarrega de fer la gestió dels usuaris. Per tal d'interaccionar-hi és important obtenir la clau API del teu servidor per tal que la petició arribi al servidor relacionat amb el projecte, en aquest cas el *AllInOneIrRemote,* ja que aquesta clau s'envia a totes les peticions HTTP que es fan. Per exemple per a crear un nou usuari (paràmetres a part)i:
https://www.googleapis.com/identitytoolkit/v3/relyingparty/signupNewUser?key=[API_KEY]

Per tal d'obtenir-la, es va a configuració del teu projecte *Firebase.*



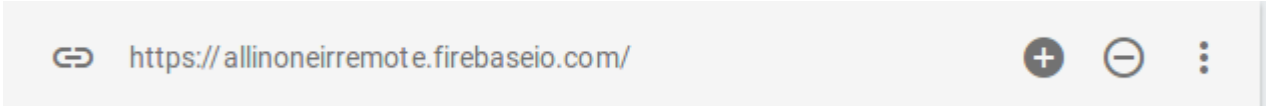
Il·lustració 39: Configuració bàsica del servidor Firebase

A més, és necessari configurar quina mètodes d'inici de sessió es permeten, en aquest cas només permetem l'inici de sessió via e-mail, ja que l'aplicació només està preparada per aquest mètode:

Il·lustració 40: Proveïdors d'inici de sessió

4.2.2. Firebase RealTime DataBase

Per a accedir a la base de dades externa cal saber on aquesta està allotjada. Es pot veure a dintre el teu projecte de *Firebase*, a l'apartat de Database:



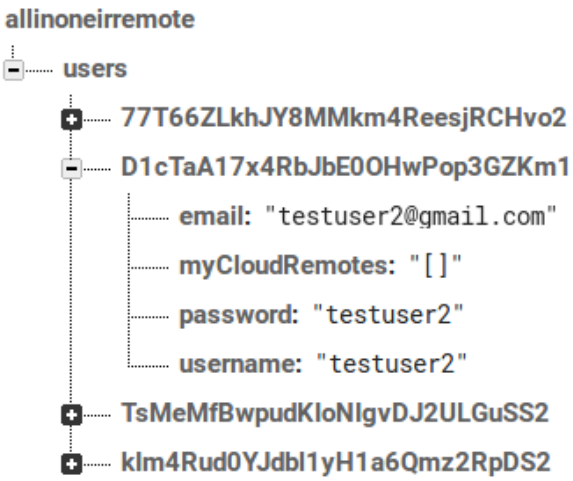
A més es pot filtrar qui escriu a la base de dades escrivint unes regles:

Il·lustració 41: Regles de la base de dades a Firebase

D'aquesta manera només es podrà escriure a la base de dades si l'usuari està amb la sessió iniciada. Per això és necessari que en la petició un paràmetre que verifiqui l'usuari. Això és fa passant *auth = idToken*. Per exemple:

```
XMLHTTP.doXMLHttpRequest("PUT", "https://allinoneirremote.firebaseio.com/users/" + localId + ".json?auth=" + idToken,'{"email":"'+ email + '","username":"'+ username + '","password":"' + password +'","myCloudRemotes":"[]"}', function (rsp) { parse(rsp)}, "application/json")
```

En aquest cas *idToken* és el paràmetre que permet verificar que l'usuari té la sessió iniciada i *localId* no és més que la ruta on s'ha d'escriure, ja que es crea un objecte per a cada usuari i el valor de localId és únic per a cada usuari:



Il·lustració 42: Visualització de la base de dades des de Firebase

5 | Cost del projecte i comercialització

En aquest apartat detallarem els costos associats al desenvolupament de l'aplicació *AllInOneIrRemote*. Els costos es poden dividir en 3 parts: costos de recursos humans, costos materials i costos addicionals.

5.1. Costos de recursos humans

El projecte s'ha realitzat amb els recursos humans formats per un enginyer/programador junior del qual suposarem una tarifa horària de 20€/hora.

Concepte	Hores Dedicades (h)	Cost/hora (€/h)	Cost (€)	Cost Acumulat (€)
Brainstorming	20	20	400	400
Estudi de l'eina d'implementació: Llenguatge Qt	10	20	200	600
Aprenentatge i/o tutorials del llenguatge de programació	60	20	1200	1800
Disseny de funcions	20	20	400	2200
Disseny de pantalles	20	20	40	2800
Implementació de l'aplicació	300	20	6000	8800
Redacció de la memòria	60	20	1200	1000
TOTAL	**500**	**-**	**-**	**10000**

5.2. Costos materials

Els costos materials que suposa aquest projecte són en la seva totalitat recursos informàtics.

Per tal de poder desenvolupar aquesta aplicació ha sigut imprescindible disposar d'un ordinador i un *smartphone* amb el sistema operatiu Android per a testejar l'app.

Així, tenint en compte l'amortització d'aquestes eines podem extreure la següent taula:

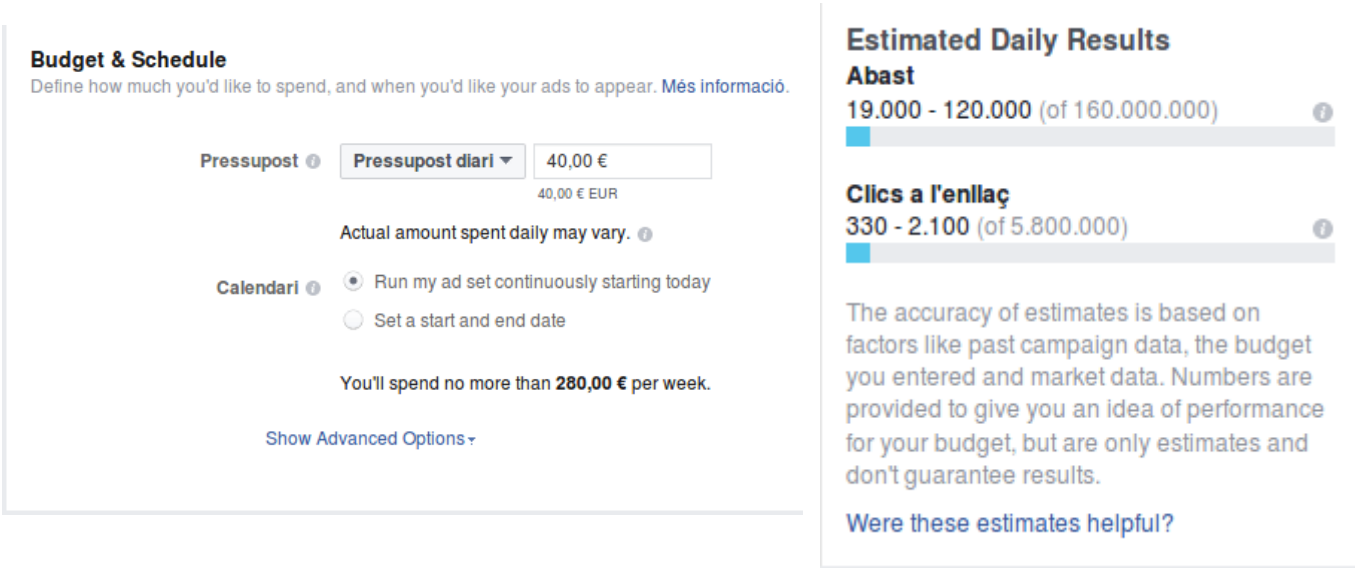
Eina	Cost (€)	Vida útil (Anys)	Duració projecte (Anys)	Cost pel projecte (€)
Lenovo Thinkpad Yoga 15	800	5	0,5	80
One Plus X	240	2	0,5	60
TOTAL	-	-	-	**140**

5.3. Costos de comercialització

Per tal que l'aplicació generi un rendiment econòmic aquesta s'ha de comercialitzar. Aquest procés també té uns costos associats.

Primer de tot si es vol penjar l'aplicació a la plataforma *Google Play* per tal que qualsevol persona se la pugui baixar. El cost per tenir un compte de desenvolupador i poder penjar aplicacions és de 25$ (~23€)[43].

Per donar a conèixer l'aplicació també és bona idea fer ús d'estratègies com anunciar l'aplicació a pagines amb un gran volum d'usuaris com Facebook. Aquest model de publicitat funciona que tu com a interessat en anunciar l'aplicació fixes un pressupost i en funció d'aquest ells afegeixen els anuncis a la seva pàgina. Lògicament com més alt sigui el pressupost, més visible serà el teu anunci. Agafant les dades de la pàgina oficial per a penjar anuncis del Facebook [44]:



Veiem que amb un pressupost setmanal de 280€ estimen que aproximadament hi haurà entre 330-2100 persones que clicaran a l'enllaç per obtenir més informació sobre l'aplicació.

Per tant seguint aquest pressupost farem una campanya que duri aproximadament unes 4 setmanes per començar a promocionar l'aplicació. El cost d'aquesta campanya serà de 1120€.

Considerant aquest primer pla de comercialització molt inicial i genèric els costos seran:

Ítem	Cost (€)
Compte a Google Play	23
Anuncis a Facebook	1120
TOTAL	**1143**

5.4. Cost total

Fent les consideracions del 3 apartats anteriors tenim que el cost total del projecte puja 11283 €.

Concepte	Valor (€)
Costos RRHH	10000
Costos Materials	140
Costos de comercialització	1143
TOTAL	**11283**

5.5. Estratègies de retorn

Com es pot veure ens els dos apartats anteriors, el desenvolupament del projecte suposa uns costos associats els quals es pretenen trobar estratègies per aconseguir el seu retorn i un cert benefici si és possible.

El cost total del projecte s'ha estimat a uns 11283 €.

Tot seguit descriurem possibles estratègies per tal de recuperar la inversió.

5.5.1. Llicència

Una possible estratègia per tal que l'aplicació generés uns certs beneficis econòmics seria que per a tenir totes les funcionalitats de l'aplicació l'usuari hagués de pagar una llicència.

Es podria implementar que per tal de poder donar-se d'alta a l'aplicació, és a dir, registrar-se, l'usuari hagués de tenir una llicència. D'aquesta manera totes les funcions associades al compte, com podria ser emmagatzemar al núvol tot un seguit de comandament per tal de poder-los recuperar posteriorment des de qualsevol altre dispositiu, només serien accessibles pels usuaris que paguessin una llicència.

Estimant el preu d'aquesta llicència a 1€ (preu acord amb els preus de les aplicacions en el mercat actual) faria falta vendre més de 11.000 llicències per tal de recuperar la inversió, una quantitat molt ambiciosa.

5.5.2. Publicitat

Finalment, oferir publicitat de qualsevol tipus a l'aplicació també seria un recurs útil per tenir una font d'ingressos a l'aplicació.

Tot i això a l'intentar mantenir una estètica neta i agradable a l'aplicació, no es considera una eina que interessi pel projecte.

5.5.3. Model mixt: Publicitat/Llicència

Un altra possibilitat que és interessant d'estudiar és la d'un model mixt, de tal manera que si l'usuari vol gaudir de l'aplicació de manera gratuïta, en aquesta apareguin anuncis.

S'oferiria també la possibilitat de pagar una llicència que eliminaria els anuncis de l'aplicació per aquells usaris que així ho desitgessin.

6 | Treball futur i possibles millores

El resultat d'aquest reball és una primera versió de l'aplicació, i per tant,existeixen moltes millores que es podrien aplicar. Tot seguit, descriurem les que es consideren més importants o aquelles les quals si disposéssim de més temps s'aplicarien.

- **Millorar la gestió d'usuaris:** Sobretot el fet que actualment l'aplicació no permet editar les credencials ni les dades de l'usuari.

- **Estratègies de retorn de la inversió:** Tal i com es descriu a l'apartat 5.5, si es vol que l'aplicació generi un rendiment econòmic, faria falta que alguna de les estratègies de retorn de la inversió es duguessin a terme.

- **Evitar sobreescriure els comandaments a la base de dades externa:** Actualment, quan es pugen els comandaments emmagatzemats en local a la base de dades externa, es fa una còpia literal del que hi ha a la base de dades local i s'escriu a la base de dades externa eliminant doncs els possibles comandaments que ja hi podien haver.

- **Evitar sobreescriure els comandaments a la base de dades local:** El mateix passa quan "baixem" els comandaments de la base de dades externa al dispositiu, els comandaments en local que hi havia en aquell moment desapareixen.

- **Fer l'aplicació compatible amb qualsevol dispositiu Android:** Aquest millora, seria una de les més importants. Ja que actualment aquesta aplicació només funciona amb aquells dispositius Android que incorporen un emissor infraroig, i, avui en dia suposen menys d'un 10% del mercat de dispositius Android venguts. La idea seria crear algun petit dispositiu o *gadget* que es connectés mitjançant usb o el connector aux i tingués un emissor infraroig que fos el que enviés realment les trames. Aquest afegit faria l'aplicació molt més interessant, ja que es podria abastar un mercat molt més ampli.

- **Publicar-la al Google Play:** Per tal que qualsevol usuari d'un dispositiu Android pugui fer ús de l'aplicació és necessari publicar-la al Google Play.

7 | Competència Genèrica: Autoaprenentatge

L'aprenentatge autònom ha sigut la competència genèrica involucrada en aquest projecte. I és que malgrat partir d'un cert coneixement previ de les eines per a poder desenvolupar una aplicació per a Android, m'han fet falta molts nous coneixements per poder programar totes les funcionalitats desitjades.

Per sort, el món de la programació és un dels àmbits acadèmics on és més fàcil dur un procés d'autoaprenentatge ja que sovint a Internet pots trobar molts recursos, fòrums i comunitats de desenvolupadors, etc. que faciliten el procés.

Concretament els recursos utilitzats han sigut els següents:

- Qt Documentation[45]: Pàgina oficial de tota la documentació de l'entorn Qt. Al llarg de tot el desenvolupament s'ha fet ús de la informació penjada en aquesta pàgina. A part de les explicacions de tots els elements i classes, és de molt interès una secció dedicada a exemples i tutorials[46].

- Qt Forum[47]: Fòrum oficial de l'entorn Qt. De gran ús quan alguna explicació de la documentació no acaba de quedar clara ja que hi pots trobar problemes que ha tingut altre gent i veure com la comunitat ha proposat maneres de resoldre'ls. També en alguns moments s'ha sigut un agent actiu del fòrum preguntant a la comunitat sobre alguna tema.

- Altres fòrums: A la xarxa, hi molts altres fòrums on diferents usuaris fan preguntes i altres les responen. Malgrat no ser els oficials per l'entorn Qt, són de gran interès. Per posar un exemple, un d'aquests fòrums és l'*StackOverflow*[48].

- Tutorials YouTube: Eina molt útil, ja que vas veient pas per pas com es fa el desenvolupament sobre l'objectiu que vols aconseguir.

- Altres eines online: Internet està ple d'altres recursos que s'han anat consultant i han ajudat a poder tirar endavant el projecte com podrien ser blogs, articles, etc.

Alguns dels continguts més importants que he hagut de prendre són:

FUNCIONALITATS ENTORN QT

Per tal de poder desenvolupar l'aplicació i que aquesta complís amb els requisits, s'han descobert moltes funcionalitats que es desconeixien o no se'n sabia fer ús de l'entorn Qt.

Algunes dels aprenentatges més destacats són:

- L'eina QTranslator per tal d'oferir l'aplicació en múltiples idiomes.
- Estendre les funcionalitats del llenguatge QML creant objectes des de la banda C++ i accedint a ells.
- L'ús del QtQuick Controls 2.0 i del disseny Material.
- La creació de classes Java i la seva interacció des de la banda QtC++.
- L'ús d'icones com si fossin caràcters.
- L'ús de base de dades SQLite en local.

ACCÉS A API's EXTERNES

Ha calgut entendre com comunicar-se amb les API's externes que ens podien donar informació rellevant, en el cas de l'aplicació del projecte l'API de http://irdb.tk/. Concretament per a poder fer ús d'aquest servei, tal i com s'ha explicat al llarg de l'aplicació ha fet falta "hackejar" l'api per tal de poder obtenir els codis, ja que era un servei que no estava disponible. Gràcies a l'analitzador de xarxes *Wireshark* es va poder imitar el comportament de la web i aconseguir els codis. Tot aquest procés també va suposar un aprenentatge molt interessat que en un futur de ben segur que em serà d'utilitat.

Per altra banda també s'han fet servir els serveis de *Firebase*, per a fer la gestió d'usuaris (creació, accés i supressió de comptes) i per implementar la base de dades externa (afegir nous elements, editar-los, eliminar-los). És una eina que ofereix un munt de possibilitats malgrat que pel present projecte només haguem fet ús de dues d'aquestes. Entendre com a interactuar amb aquest servei de Google ha sigut molt interessant ja que fa que tant la base de dades externa com la gestió d'usuaris siguin molt robusts.

8| Conclusions

El projecte compleix amb les expectatives i requeriments que van ser proposats a l'inici: tenir una primera versió operativa d'una aplicació Android que permetés simular un gran nombre de comandaments a distància. A més l'aplicació resultant conté una interfície molt neta i fàcil d'utilitzar. Per això penso, que la millor conclusió és obrir l'aplicació, fer ús d'aquesta i valorar-la.

Concretament fent un anàlisis dels objectius que es volien assolir al iniciar al projecte (apartat 1.2):

- **Seleccionar el model de control remot que vol simular:** Objectiu assolit en la seva totalitat.
- **Mostrar el comandament a la pantalla del dispositiu mòbil:** Objectiu assolit en la seva totalitat
- **Emetre les mateixes trames infraroges que el comandament desitjat:** Objectiu assolit malgrat que pot ser que l'aplicació no disposi de certes funcionalitats ja que estem condicionats per la base de dades des d'on obtenim les especificacions de cada model.
- **Guardar diferents controls remots per a fer-ne ús posteriorment:** Objectiu assolit en la seva totalitat.
- **Tenir un sistema d'usuaris per tal de poder guardar els comandaments a la xarxa:** Objectiu assolit gràcies als serveis de Firebase.
- Tot els requisits funcionals i no funcionals han sigut assolits amb èxit (apartat 2.1).

En l'aspecte personal, el procés de desenvolupament de l'aplicació m'ha servit per a consolidar i aprofundir coneixements del món de la programació i, més concretament, de la programació de dispositius Android. El treball d'autoaprenentatge ha sigut constant i elevat. Ha fet falta buscar molta informació i exemples de com realitzar les diferents funcions que es volia que l'app dugués a terme i que es desconeixia la manera d'implementar. El món de la programació és probablement el millor per a poder trobar recursos a Internet i he anat podent solucionar tots els problemes amb èxit.

9 | Referències

1: Xiaomi Web Page: http://www.mi.com/en/, Juny 2017

2: GIMP Webpage: https://www.gimp.org/, Juliol 2017

3: irdb: http://irdb.tk/, Juny 2017

4: API irdb: http://irdb.tk/api, Juliol 2017

5: Fòrum irdb: http://irdb.tk/discuss, Juliol 2017

6: Wireshark Web Page: https://www.wireshark.org/, Juliol 2017

7: IDC: Smartphone OSMarket Share, 2016 Q3: http://www.idc.com/promo/smartphone-market-share/os, Juny 2017

8: Android Studio Home Page: https://developer.android.com/studio/index.html, Juny 2017

9: Qt Creator Web: https://www.qt.io/ide/, Juny 2017

10: Wikipedia: Qt Creator: https://es.wikipedia.org/wiki/Qt_Creator, Juny 2017

11: Qt QML Documentation Page: http://doc.qt.io/qt-5/qtqml-index.html, Juny 2017

12: QML Applications Documentation Page: http://doc.qt.io/qt-5/qmlapplications.html, Juny 217

13: Ubuntu QML Apps Web Page: https://developer.ubuntu.com/en/phone/apps/qml/, Juny 2017

14: Wikipedia: QML: https://es.wikipedia.org/wiki/QML, Juny 2017

15: SQLite Web: https://www.sqlite.org/, Agost 2017

16: SQLite on QML: http://doc.qt.io/qt-5/qtquick-localstorage-qmlmodule.html, Agost 2017

17: Firebase Google: https://firebase.google.com/, Agost 2017

18: FIrebase REST API: https://firebase.google.com/docs/reference/rest/database/, Agost 2017

19: Sublime Text Web: https://www.sublimetext.com/, Juliol 2017

20: Moqups Web Page: https://app.moqups.com, Juny 2017

21: GenyMotion Web: https://www.genymotion.com/, Juny 2017

22: Qt: Material Design: https://doc.qt.io/qt-5/qtquickcontrols2-material.html, Juny 2017

23: Material Design: https://material.io/guidelines/, Juny 2017

24: Material Icons: https://material.io/icons/, Juny 2017

25: Firebase REST API Authentication: https://firebase.google.com/docs/reference/rest/auth/, Agost 2017

26: Font Awesome: http://fontawesome.io/, Juny 2017

27: Material Design Icons TTF: https://github.com/google/material-design-icons/blob/master/iconfont/MaterialIcons-Regular.ttf, Juliol 2017

28: How To Translations (i18n): https://appbus.wordpress.com/2016/04/28/howto-translations-i18n/, Juliol 2017

29: Qt Documentation: Internationalization and Localization with Qt Quick: http://doc.qt.io/qt-5/qtquick-internationalization.html, Juliol 2017

30: QTranslator Class: http://doc.qt.io/qt-5/qtranslator.html, Juliol 2017

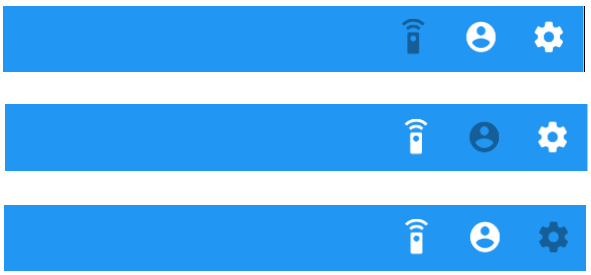
31: Qt Wiki: How to do dynamic translation in QML: https://wiki.qt.io/How_to_do_dynamic_translation_in_QML, Juliol 2017
32: Qt Documentation: Embedding C++ Objects into QML with Context Properties: http://doc.qt.io/qt-5/qtqml-cppintegration-contextproperties.html, Juliol 2017
33: Qt Documentation: Defining QML Types from C++: http://doc.qt.io/qt-5/qtqml-cppintegration-definetypes.html, Juliol 2017
34: Qt Documentation: Writing QML Extensions with C++: http://doc.qt.io/qt-5/qtqml-tutorials-extending-qml-example.html, Juliol 2017
35: Android->StatusBar: https://developer.android.com/reference/android/view/Window.html#setStatusBarColor(int), Juliol 2017
36: StackOverlow -> Change StatusBar Color: https://stackoverflow.com/questions/22192291/how-to-change-the-status-bar-color-in-android, Juliol 2017
37: Qt Android Extras: http://doc.qt.io/qt-5/qtandroidextras-index.html, Juliol 2017
38: Qt AndroidJNIObject: http://doc.qt.io/qt-5/qandroidjniobject.html, Juliol 2017
39: XMLHTTP request API DOC: https://developer.mozilla.org/es/docs/Web/API/XMLHttpRequest, Juliol 2017
40: ProntoHex Code Format: http://www.remotecentral.com/features/irdisp2.htm, Agost 2017
41: Android Documentation: ConsumerIrManager: https://developer.android.com/reference/android/hardware/ConsumerIrManager.html, Agost 2017
42: QtDocumentation: Qt Quick Local Storage: http://doc.qt.io/qt-5/qtquick-localstorage-qmlmodule.html, Agost 2017
43: Google Play Developers Sign Up: https://play.google.com/apps/publish/signup/, Setembre 2017
44: Facebook Ads: www.facebook.com/ads/create, Setembre 2017
45: Qt Documentation: http://doc.qt.io/, Juny 2017
46: Qt Documentation: Qt Examples And Tutorials: http://doc.qt.io/qt-5/qtexamplesandtutorials.html, Juny 2017
47: Forum Qt: https://forum.qt.io/, Juny 2017
48: StackOverflow: https://es.stackoverflow.com/, Juliol 2017

ANNEX A: Manual d'usuari

L'aplicació es troba dividida en tres gran seccions:

- Secció comandaments.
- Secció gestió d'usuari.
- Secció configuració.

Per tal de permetre navegar-hi, a la part superior de l'aplicació es mostra en tot moment la següent barra:



Com es pot veure a la imatge, hi ha una icona que sempre està deshabilitada. Sempre que ens trobem a una secció, la icona corresponent sempre estarà inactiva.

Al arrancar l'aplicació la secció que es mostra és la que gestiona els comandaments:

1 Gestió dels comandaments

A la gestió d'usuaris, s'hi pot accedir sempre que es desitgi polsant la icona . És la secció que es mostra per defecte al iniciar l'aplicació.

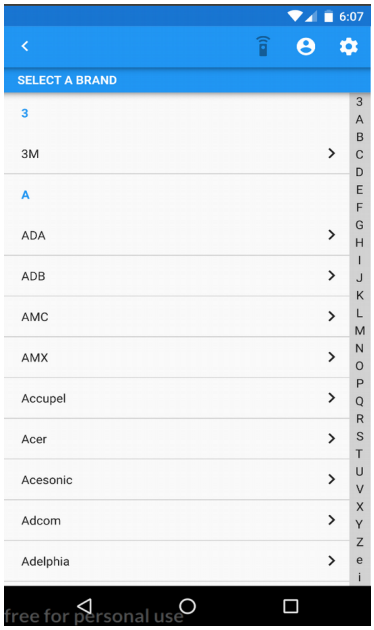
Al arrancar l'aplicació poden passar dues coses, en el cas que la base de dades local del dispositiu tingui algun comandament guardat, apareixerà la pantalla per a controlar la TV, el DVD (explicada més endavant)... Si per cas contrari no tenim comandament a la base de dades, apareixerà la següent pantalla:

Aquesta pantalla, ens invita a començar a fer el procés de generar un comandament a distància. També s'aprecia a la ActionBar superior un navegador amb tres icones:

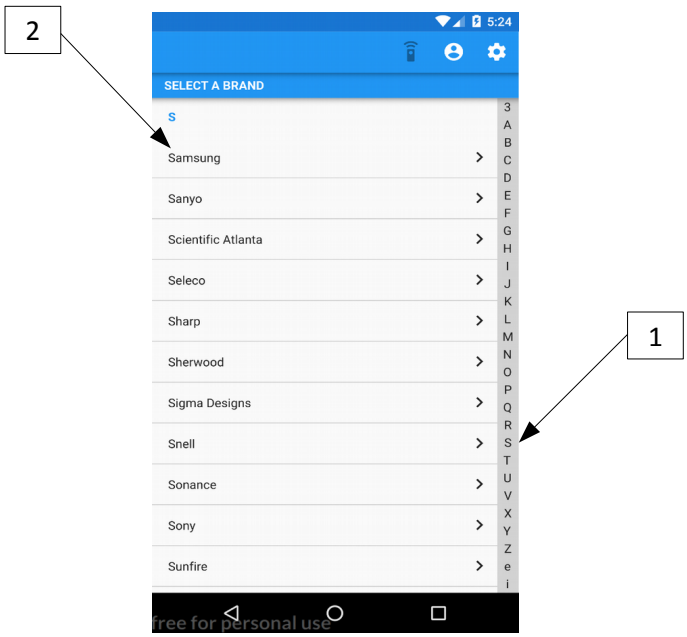
Així doncs, al polsar el botó d'afegir un nou comandament, s'inicia el procés per a seleccionar quin comandament es vol simular des de l'aplicació:

1.1. Selecció de la marca

El primer pas que cal dur a terme és escollir la marca del dispositiu.

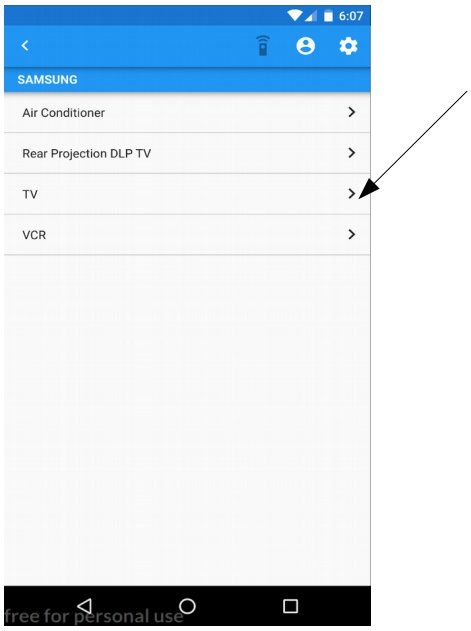


Per tal d'agilitzar el procés, a la banda dreta de la pantalla apareix una llista amb una llista de caràcters. Al polsar sobre aquest, la llista es mourà fins a la secció de la llista on les marques comencen amb el caràcter seleccionat i per anar a la següent pantalla s'ha de polsar sobre la marca desitjada.



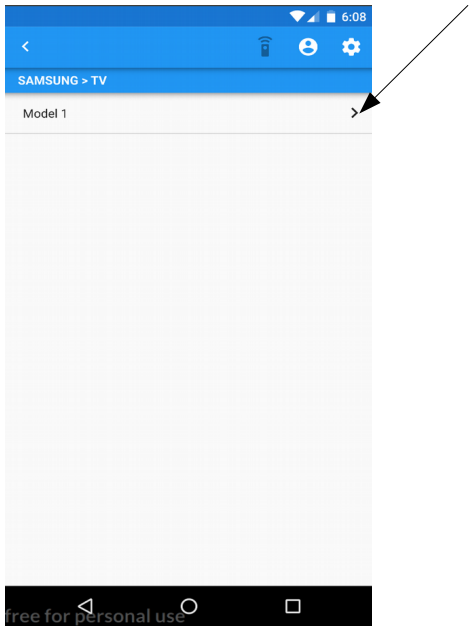
1.2. Selecció del tipus de dispositiu

Un cop seleccionada la marca, es mostren tots els tipus de dispositius dels quals tenim codis infrarojos d'aquella marca:



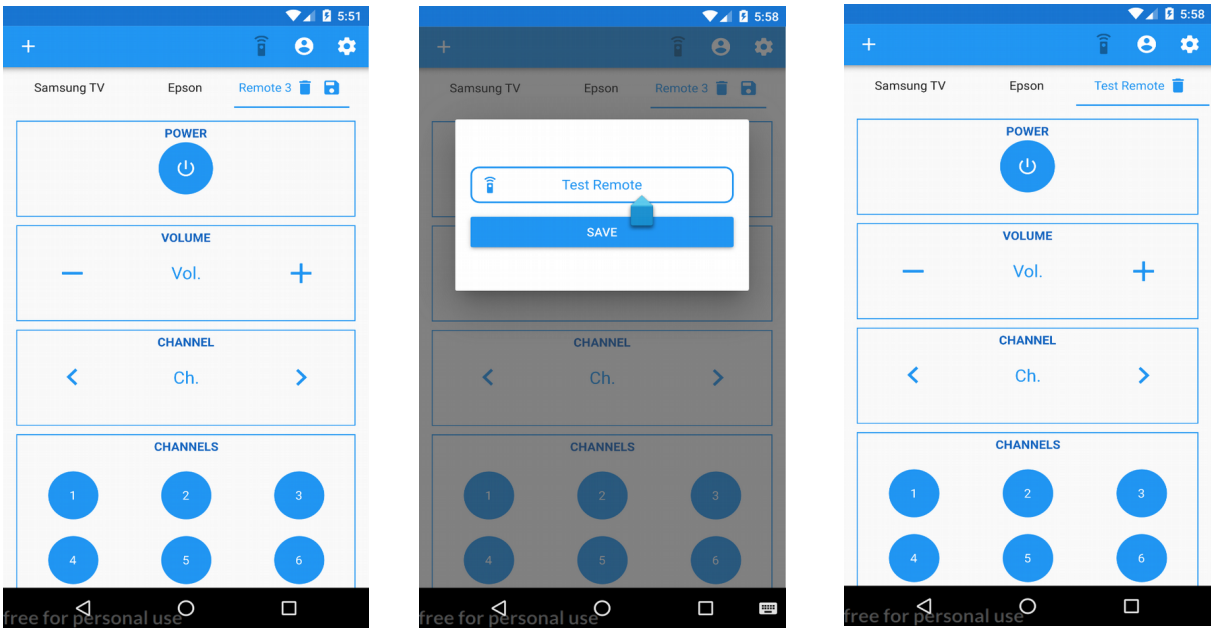
1.3. Selecció del model

Per una mateixa marca i un mateix dispositiu poden existir diferents models. És per això que cal seleccionar d'entre tots els models disponibles, aquell que funciona amb l'aparell que volem controlar. Alhora de seleccionar quin model seleccionar, la millor pràctica és començar pel primer i si aquell no funciona anar canviant fins a trobar el que controli el nostre aparell.



1.4. «Layout» del comandament

Finalment alhora de seleccionar el model, apareix la pantalla des de la qual ja podem enviar trames infraroges:



La pantalla consta de tots els botons disponibles que es disposen del comandament del qual es vol simular i simplement cal polsar sobre el botó desitjat apuntant l'aparell.

El nom del comandament per defecte és «Remote n» . Si el comandament descobert és el que volíem i el volem guardar es polsa sobre la icona de guardar i apareixerà una finestra *Popup* que ens permetrà assignar-li un nom:

Després d'assignar el nom, es polsa guardar i el comandament ja està anomenat tal i com l'usuari desitja. A més al guardar-lo, el següent cop que l'usuari obri l'aplicació, el comandament estarà disponible per a fer-lo servir.

Un cop a aquesta pantalla, si es vol tornar a iniciar el procés per configurar un nou comandament, es pot fer clicant a la icona «+» que es troba a la cantonada superior esquerra.

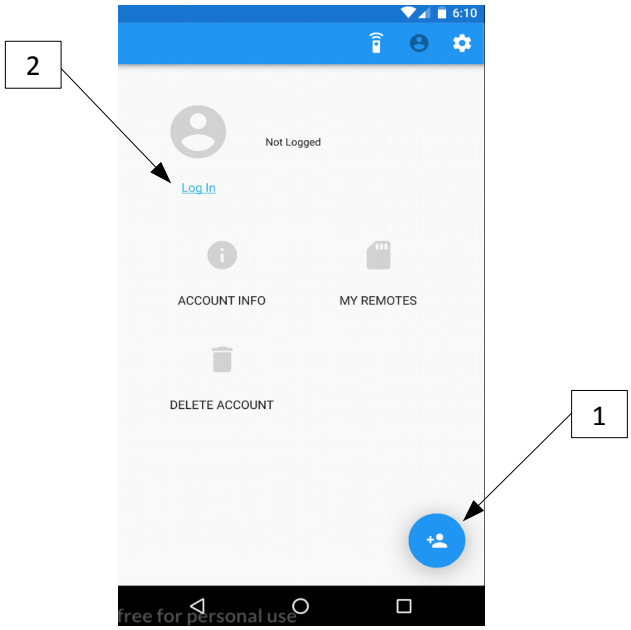
Sempre que es desitgi, qualsevol comandament pot ser eliminat polsant sobre la icona d'eliminar.

Aquesta pantalla, tal i com s'ha comentat al principi de l'apartat 1, és la que apareixerà per defecte al iniciar la sessió en el cas que tinguem com a mínim un comandament guardat.

2 Gestió d'usuaris

A la gestió d'usuaris, s'hi pot accedir sempre que es desitgi polsant la icona .

Si encara no tenim la sessió iniciada se'ns mostrarà la següent pantalla:



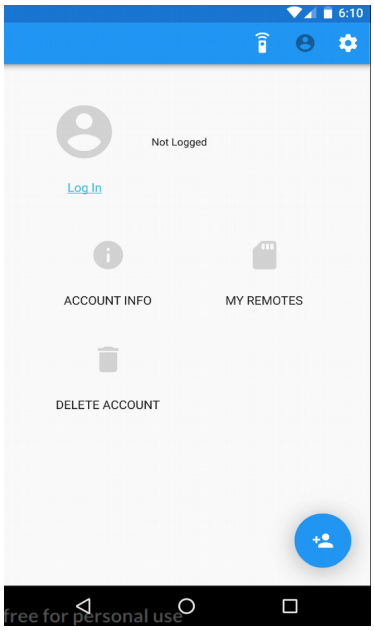
Com es pot veure, pràcticament tots els botons es troben desactivats, ja que es tracta de funcionalitats relacionades a l'usuari, i, per tant, cal que l'usuari tingui la sessió iniciada. Només tenim dos control habilitats, els que permeten:

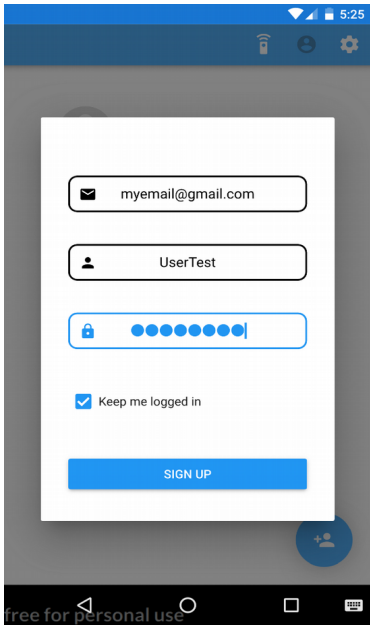
1. Registrar-se (Sign Up)

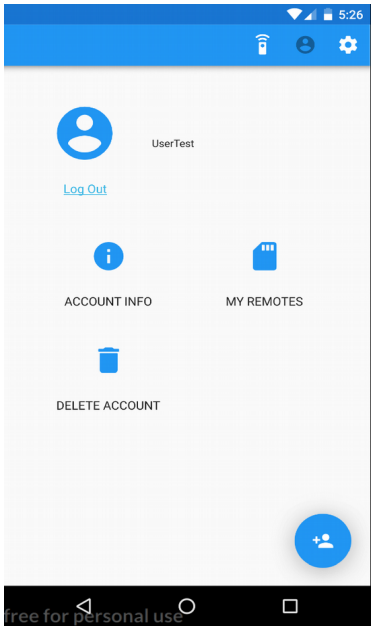
2. Iniciar Sessió (Log In)

2.1. Registrar-se (Sign Up)

Un cop es clica el botó per a registrar-se (situat a la cantonada inferior dreta de la pantalla), sobre un Pop Up amb el formulari a omplir per a crear un nou usuari:



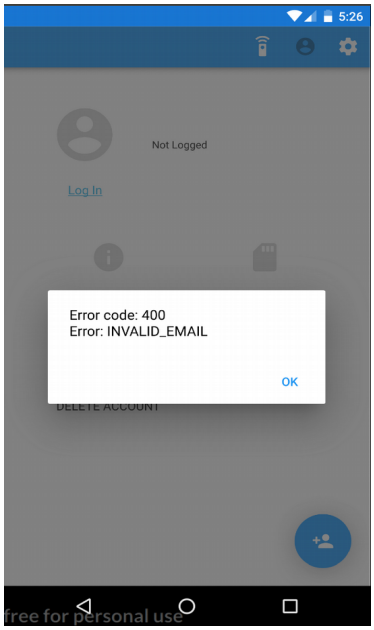




Un cop introduïts tots els camps que es demanen i després de polsar el botó «SIGN UP», si tot ha anat bé, es tornarà a la pantalla principal de la gestió d'usuaris però ara ja amb la sessió iniciada i lògicament amb l'usuari creat al servidor. Com que la sessió està iniciada, tots els controls de la pàgina ja es mostraran actius.

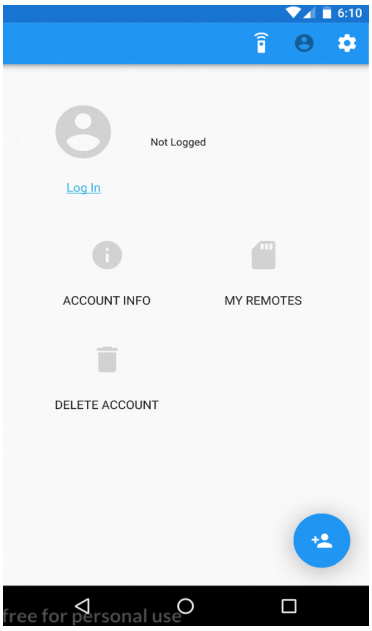
Al final del formulari per a registrar-se hi ha un *CheckBox*, al marcar-lo forcem que l'usuari que es registre i per tant el que s'iniciarà la sessió es guardi a la base de dades locals del dispositiu per tal que el següent cop que obrim l'aplicació, ja tinguem la sessió iniciada.

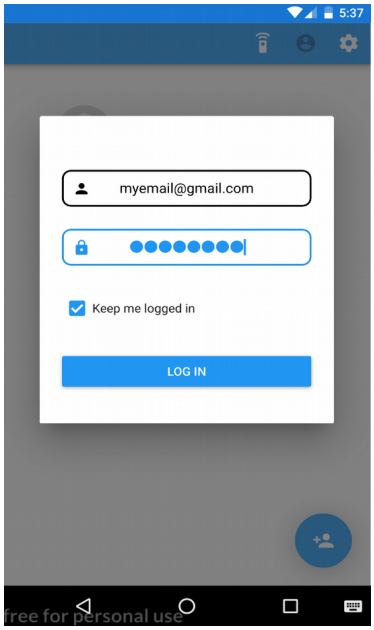
En el cas que hi hagi hagut qualsevol tipus d'error durant el procés de registrament, aquest es mostrarà en mode d'alerta i com és d'esperar, l'usuari no s'haurà creat al servidor.

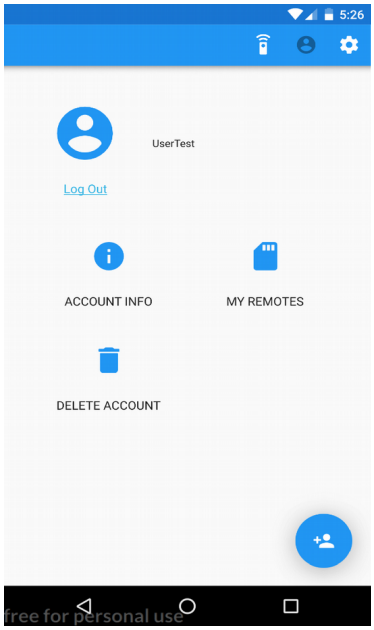


2.2. Iniciar Sessió (Log In)

Un cop es polsa sobre l'opció *Log In*, sobre de nou un Pop Up amb el formulari a omplir per tal d'iniciar la sessió:



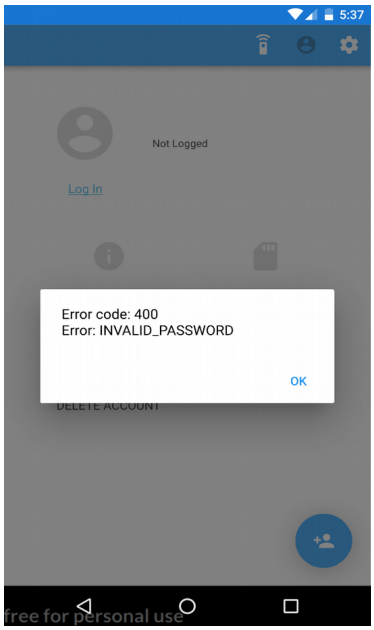




Un cop introduïts tots els camps que es demanen i després de polsar el botó «LOG IN», si tot ha anat bé, es tornarà a la pantalla principal de la gestió d'usuaris però ara ja amb la sessió iniciada. Com que la sessió està iniciada, tots els controls de la pàgina ja es mostraran actius.

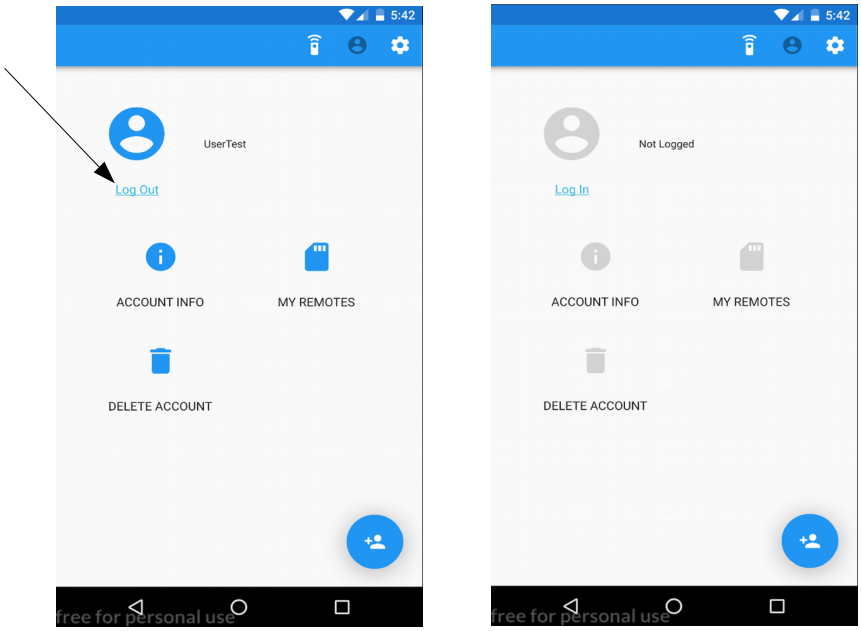
Al final del formulari per a registrar-se hi ha un *CheckBox*, al marcar-lo forcem que l'usuari que es registre i per tant el que s'iniciarà la sessió es guardi a la base de dades locals del dispositiu per tal que el següent cop que obrim l'aplicació, ja tinguem la sessió iniciada.

En el cas que hi hagi hagut qualsevol tipus d'error durant el procés de registrament, aquest es mostrarà en mode d'alerta i com és d'esperar, la sessió no estarà iniciada.



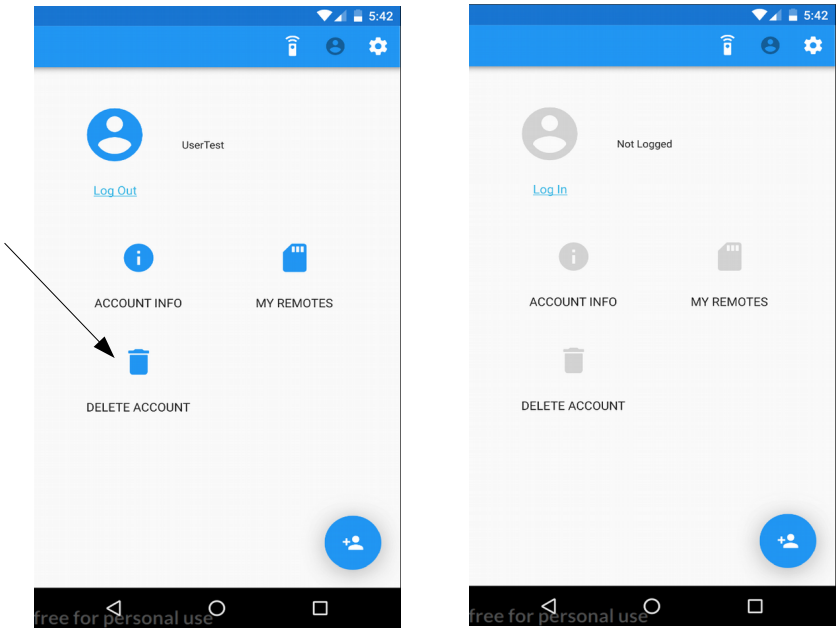
2.3. Tancar Sessió

Per tancar la sessió simplement cal polsar a *Log Out.*



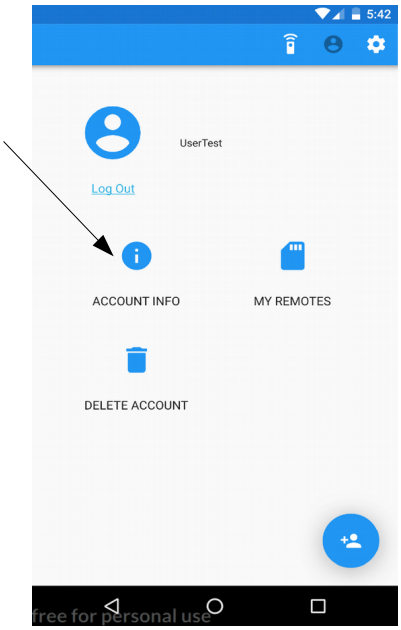
2.4. Eliminar Sessió

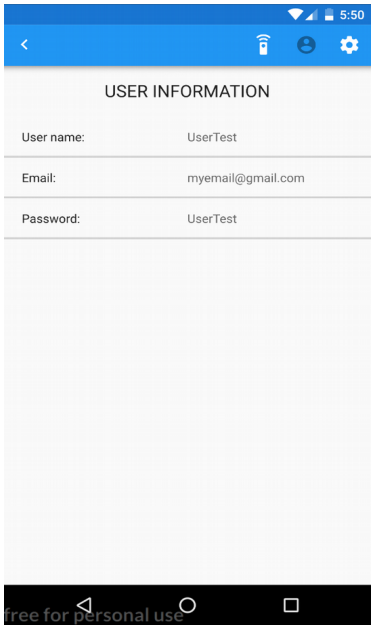
Al polsar sobre la icona eliminar, l'usuari serà eliminat del servidor i la sessió en l'aplicació es tancarà.



2.5. Informació del compte

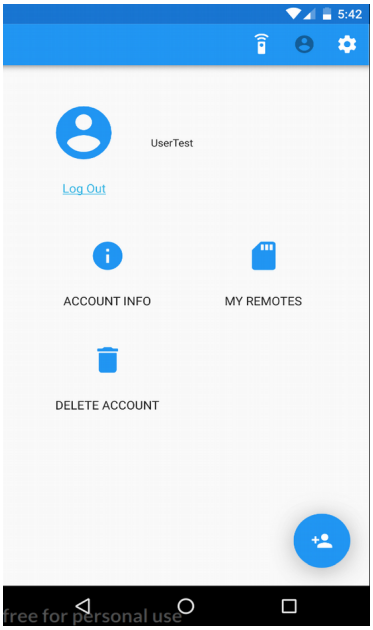
Quan ens trobem amb la sessió iniciada, tenim la possibilitat de visualitzar la informació corresponent al compte:

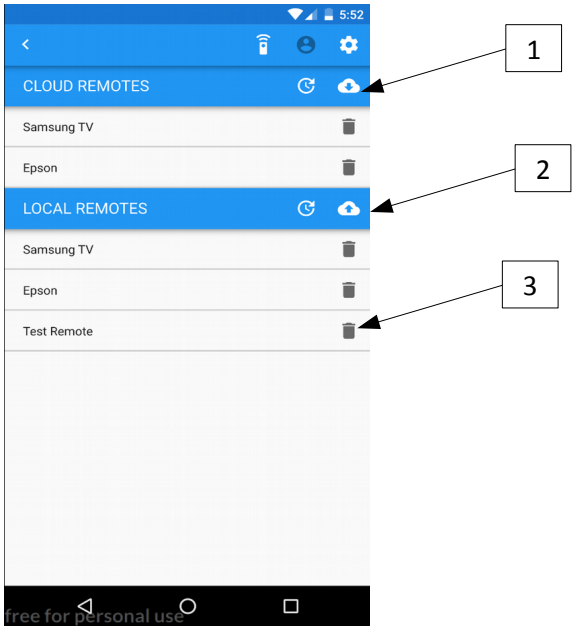




2.6. Els meus comandaments

Polsant sobre *My Remotes*, entrem a una pantalla per tal de gestionar els comandaments que tenim emmagatzemats tant a la base de dades local com a la base de dades externa associada el nostre compte:





1. Descarregar comandaments guardats a la base de dades externa a la base de dades locals

   Al polsar (1), tots els controls que estan associats al nostre usuari, seran descarregats a la base de dades local i per tant, podran ser utilitzats per a l'aplicació.

2. Pujar comandaments a la base de dades externa

   Al polsar (2), tots els comandaments que es trobin a la base de dades local, seran pujats a la base de dades externa associada el nostre compte per tal de poder recuperar-los més endavant des de qualsevol altre dispositiu.

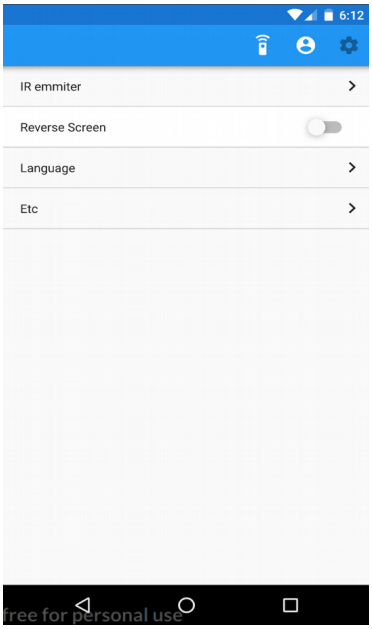
3. Eliminar comandament

   Al polsar (3), el comandament associat a la icona serà eliminat ja sigui de la base de dades local o de la base de dades externa (en funció del comandament que estiguis apuntant).

3 GESTIÓ DE LES PREFERÈNCIES

A la gestió d'usuaris, s'hi pot accedir sempre que es desitgi polsant la icona  .

La pantalla que es mostra és la següent:



De moment només està operativa la funcionalitat de les preferències d'idioma. L'elecció de l'emissor infraroig, així com la de capgirar l'orientació de la pantalla encara no estan implementades.

Al entrar a les preferències d'idioma, podrem escollir un dels tres idiomes en els quals l'aplicació està disponible, català, castellà o anglès. L'idioma en què l'aplicació es mostrarà al iniciar-se serà la que detecti que estigui fent servir el sistema operatiu.